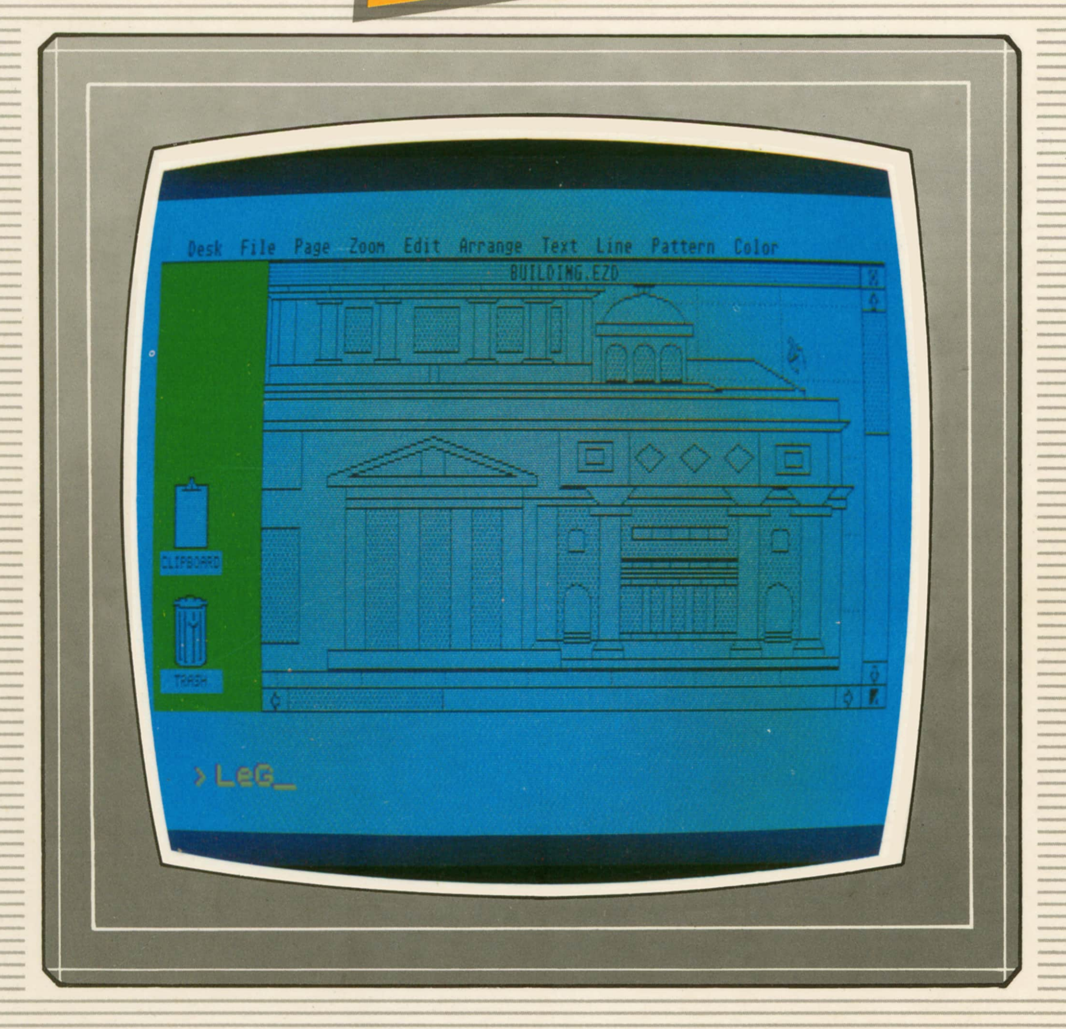
▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 38

PASO A PASO



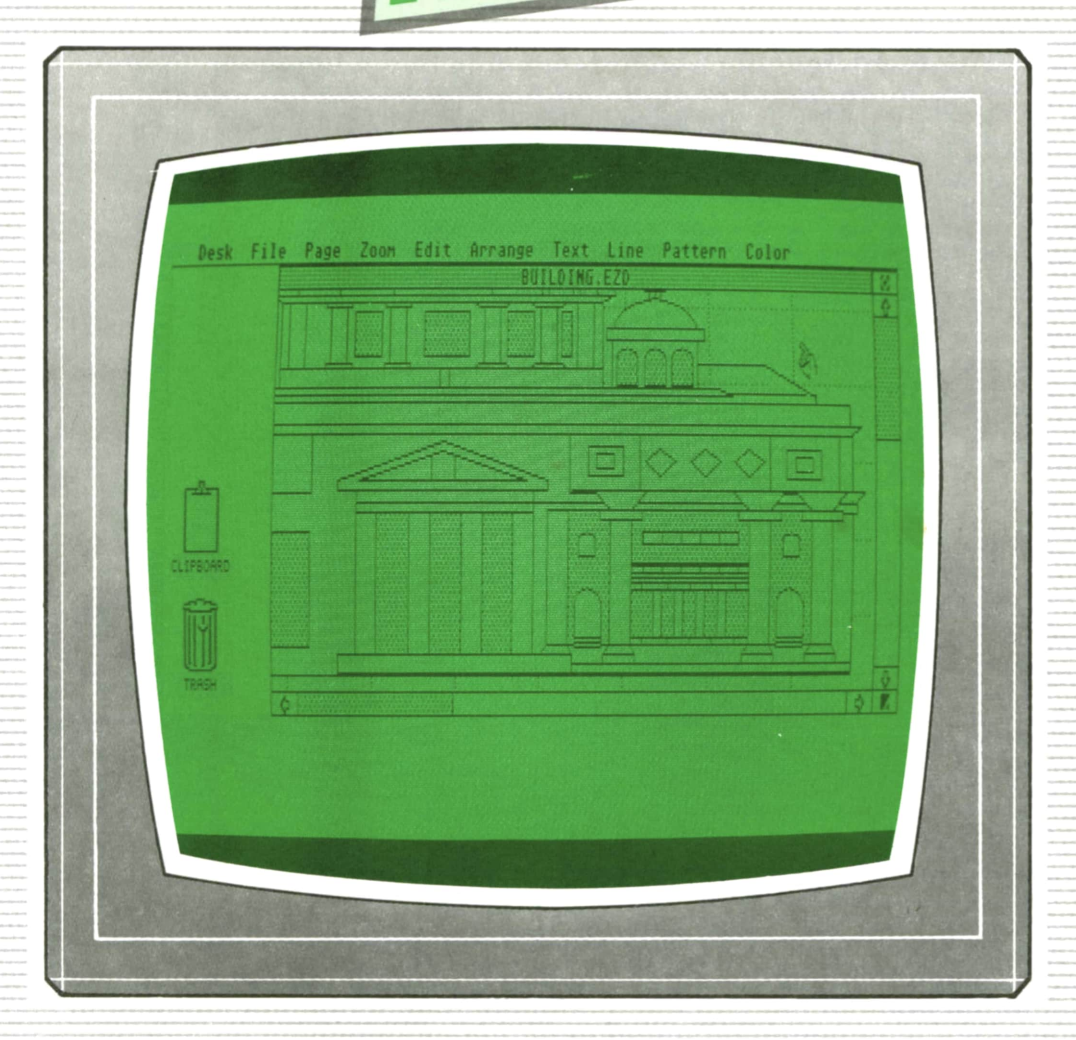
PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 38

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.
Gerente:
ANTONIO G. CUERPO.
Directora de producción:
MARIA LUISA SUAREZ PEREZ.
Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.
Diseño y maquetación:
BRAVO-LOFISH.
Fotografía:
EQUIPO GALATA.
Dibujos:
JOSE OCHOA

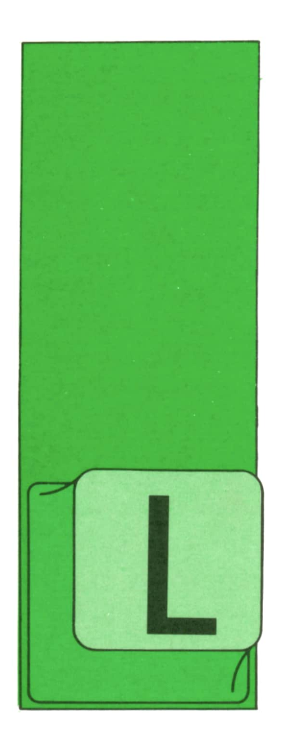
TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas, Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de Aula de Informática Aplicada (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanera, Licenciada en Informática. APLICACIONES: Jorge Thomas, Técnico en Informática. OTROS LENGUAJES (ADA): Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación.

Ediciones Siglo Cultural, S.A.
Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 455 09 99. 28020 Madrid.
Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.
Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.
Distribución en Ecuador: Muñoz Hnos.
Distribución en Perú: DISELPESA.
Distribución en Chile: Alfa Ltda.
Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-186-X
ISBN de la obra: 84-7688-068-7
Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.
Imprime:
MATEU CROMO. Pinto (Madrid).

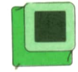
Depósito legal: M. 5.677-1987
Printed in Spain - Impreso en España.
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 259 73 31. 28020 Madrid.
Abril, 1988
P.V.P. Canarias: 335,–.

INDICE



INFORMATICA BASICA

LA TRANSMISION DE DATOS



Transmisión. Tipos

A transmisión de datos consiste en el transporte físico de la información de un lugar remoto a otro, como puede ser de un ordenador a un terminal y viceversa.

TRANSMISION DE DATOS

Ambito:

Facilidades de transmisión de datos.
Técnicas de transmisión.
Control de cambio.
Implicaciones para el usuario de un sistema de telegestión.

DEFINICION Y EJEMPLOS DE TRANSMISION DE DATOS

Definición:

La reproducción de la información en un lugar remoto.

Ejemplos:

Envío de una carta.
Llamada telefónica.
Télex.
Proceso de datos por ordenador.

La vía de transmisión puede incluir elementos como:

- Controladores de líneas.
- Líneas de comunicaciones.
- Concentradores y multiplexores.
- Modems.

Existen dos formas básicas de transmisión.

1. *Transmisión asíncrona*

Es un modo de transmisión en el que se asocia a cada carácter un bit de arranque y otro de parada, de forma que el receptor y el emisor están sincronizados durante el tiempo que dura la transmisión del carácter. La transmisión asíncrona es idónea para sistemas que funcionan a baja velocidad.

2. *Transmisión síncrona*

En este modo de transmisión, el emisor y receptor están sincronizados durante todo el tiempo que dura la transmisión del mensaje. La transmisión síncrona es más conveniente cuando la velocidad es superior a 1.200 bps y se quiere transmitir grandes volúmenes de información.

START	Carácter	STOP	START	Carácter	STOP	



Fig. 1. **TRANSMISION ASINCRONA**

SINC	Bloque	SINC	Bloque



Fig. 2. **TRANSMISION SINCRONA**



Tipos de transmisión

Podemos distinguir entre:

— Sistemas de transmisión en línea (ON-LINE). Son sistemas que envían los datos directamente al ordenador.

— Sistemas de transmisión fuera de línea (OFF-LINE), que utilizan memorias auxiliares para almacenar los datos, que después serán utilizados por el ordenador.

— Sistemas interactivos, que admiten que el usuario "dialogue" con el ordenador.

— Sistemas no interactivos, que procesan las órdenes dadas por el usuario por lotes, sin permitir diálogos mientras se ejecutan.

En toda transmisión se pueden producir errores, haciendo que la información

emitida no coincida con la recibida. Ello es debido a distintas causas:

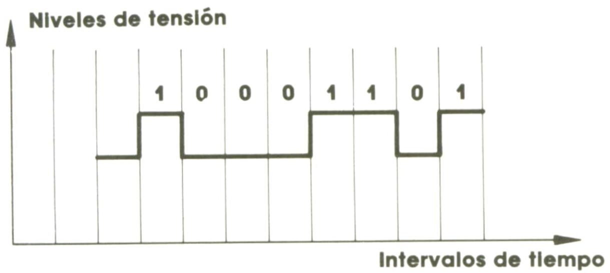
— Líneas de transmisión. Las propiedades eléctricas de las líneas de transmisión, como son la capacitancia, resistencia e inductancia, hacen que los datos se distorsionen por el camino.

— Ruido térmico. En todos los circuitos se producen constantemente vibraciones en sus átomos y moléculas que aumentan con la temperatura. El motivo de que se produzca ruido es que los átomos a medida que vibran producen ondas electromagnéticas que originan "un ruido de fondo".

— Velocidad de las líneas. Si la velocidad de transmisión es lenta se producen menos distorsiones, de forma que si:

— Se transmiten dos impulsos por segundo, el mensaje podría ser recibido perfectamente por el ser humano;

— Se transmiten 10 impulsos por segundo, sólo podría ser recibido por un equipo receptor muy sensible.



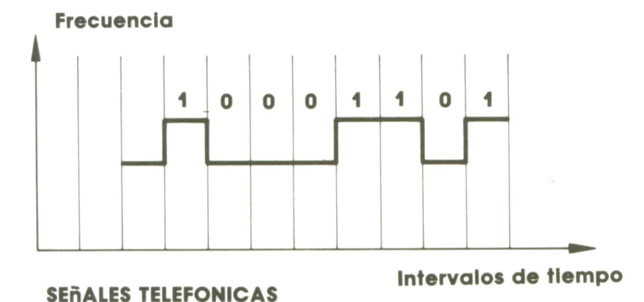



Fig. 3.

Para proteger los mensajes de posibles errores de transmisión existen diversos métodos.

1. *Paridad de caracteres*

Consiste en añadir un bit extra (bit de paridad) a cada carácter.

2. *Paridad de bloques*

Cuando un número de caracteres se transmite como un bloque, cada bloque puede llevar un carácter adicional de control.

3. *Realimentación de la información*

Se transmite un duplicado de la información recibida del receptor al emisor.

4. *Códigos especiales*

Aumentan la posibilidad de detección de errores, corrigiéndolos en algunos casos sin tener que retransmitir.

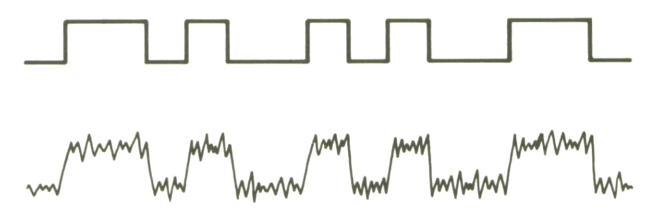



Fig. 4. Correspondencia entre una señal analógica y digital.



Vías de transmisión

Hay tres vías de transmisión:

— Vía de tipo telefónico.
— Vía de tipo telegráfico.
— Vía de tipo banda ancha.

La diferencia radica principalmente en la velocidad de transmisión. La clasificación se hace de acuerdo a:

— Baja velocidad: Son las líneas que transmiten entre 50 y 200 bps, equivalente a transmitir de tres a 20 caracteres por segundo.

— Velocidad media: Transmisión entre 600 y 4.800 bps, es decir, entre 20 y 100 caracteres por segundo.

— Alta velocidad: Transmiten entre 9.600 y 48.000 bps, equivalente a transmitir entre 300 y 90.000 caracteres por segundo.

Cuando varios usuarios están utilizando a la vez la misma línea de comunicación, la independencia que existe entre ellos depende mucho de dicha línea. Con algunas líneas todos los usuarios deben utilizar el mismo terminal con el mismo procedimiento de control de línea. Con otras líneas, los terminales pueden ser de

distinto tipo, pero deben utilizar la misma clase de caracteres de control. Otras llegan a permitir que los terminales sean completamente independientes, e incluso que utilicen distintas clases de caracteres.



Capacidad de una línea de transmisión

La capacidad de una línea de comunicación se mide mediante la relación de la cantidad de datos transmitidos y el tiempo empleado en transmitir dichos datos. A veces se necesita que la cantidad de datos transmitidos sea muy grande y en otros casos simplemente se requiere la transmisión de un bit de control. En cuanto al tiempo de transmisión, unas veces exige una respuesta rápida y en otros la respuesta del ordenador al usuario puede retardarse un poco más.

El tiempo de respuesta por parte del ordenador también varía con la aplicación que se esté ejecutando.

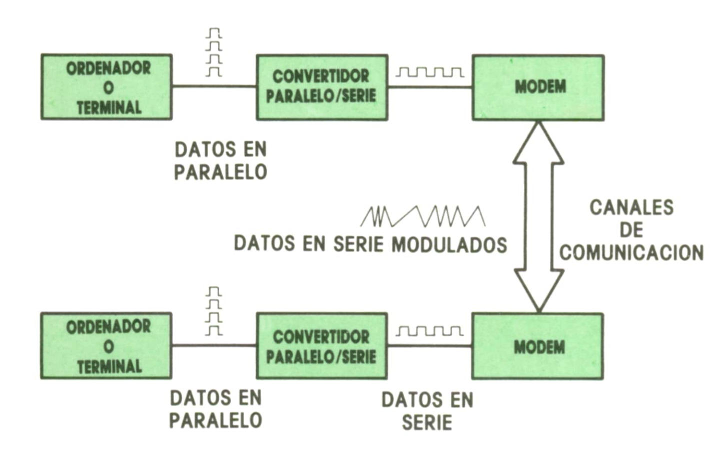
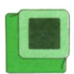




Fig. 5. Elementos que componen un sistema de comunicación.



Líneas telefónicas

Para poder utilizar la infraestructura de la línea telefónica en la transmisión de datos entre el ordenador y los terminales se ha utilizado la transmisión digital.

Como sabemos, la información fluye entre ordenadores, y dentro de ellos mismos, en forma de señales analógicas. Para convertir la corriente de "bits cuadrados" en una gama de frecuencias apropiadas para poder transmitirlos mediante la línea telefónica se utilizan los *modems.*



Modems

Para que los datos puedan "viajar" de unos ordenadores a otros utilizando como medio de transmisión la línea telefónica, es necesario realizar una transformación de señales. La función de los modems consiste en realizar estas transformaciones.

Internamente el ordenador trabaja con datos agrupados de ocho en ocho, dieciséis en dieciséis, etc., dependiendo del tamaño de la "palabra". Todos estos bits agrupados se manejan a la vez, es decir, los ocho o dieciséis de la palabra se mueven juntos en formato paralelo. Esto se hace así debido a que aumenta la velocidad de proceso.

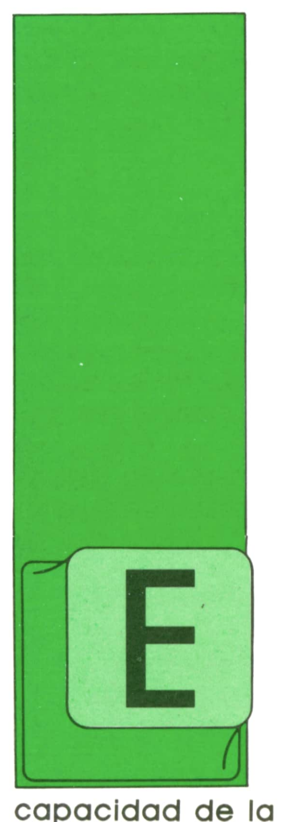


Fig. 6. El modem: permite comunicación entre ordenadores, a través de la línea telefónica.

Sin embargo, cuando lo que se pretende es que los datos se transmitan, el formato no suele ser en paralelo, sino en serie, debido a que el costo de enviar datos en paralelo es mucho mayor al necesitar un hilo para cada dato transmitido.

Cuando los datos se transmiten en serie van unos detrás de otros, utilizando una misma línea. Cuando un ordenador quiere comunicarse con otro, enviándole datos, el modem toma esta información y transforma todas las señales binarias que quiere enviar el ordenador, de forma que primero hace la conversión de paralelo a serie. Luego Modula las señales haciendo un cambio digital-analógico y envía esta señal por la línea telefónica. Al llegar al punto de destino se tiene que producir la operación contraria, es decir, al otro extremo de la línea tiene que haber otro modem que DEMOdule la señal convirtiéndola de nuevo en información propia del ordenador para poder ser entregada al ordenador receptor.

MAQUINA 6502



Algunas rutinas útiles

EN este último capítulo de los que se han dedicado al funcionamiento del mismo procesador 65XX incluiremos algunas rutinas interesantes que aprovechan la capacidad de la ROM del sistema operativo. Se dará una breve explicación de su funcionamiento y a continuación el listado de la rutina. Espero que sean de utilidad y que puedan ayudar a escribir otras basándose en ellas.

Modificación de la función LIST

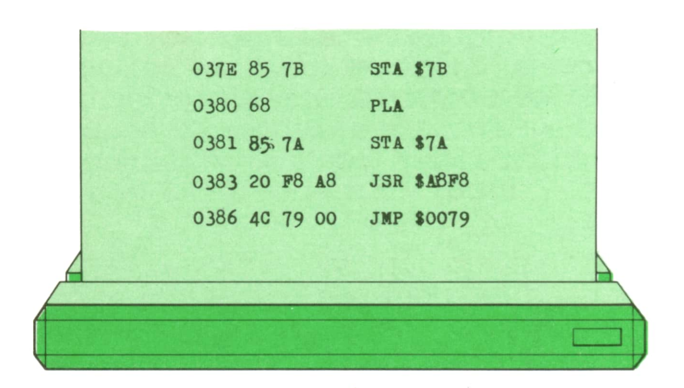
Si tratamos de incluir la función LIST en un programa BASIC obtendremos indudablemente un listado del programa en memoria, pero después se retornará al modo directo sin poder proseguir el programa que se estaba ejecutando. Asimismo, no podemos listar una línea o listar X veces el programa con un bucle.

Esto puede solucionarse cambiando el final de la rutina "LIST" que incluye un "arranque" del BASIC por un comando RTS. Además, se deberá guardar el apuntador al texto del programa donde se incluyó el comando LIST, para que luego el programa pueda proseguir donde se interrumpió.

He evitado el introducir etiquetas para aquellos que no puedan introducirlas en su programa monitor. A continuación está el listado del programa. Una vez introducida la rutina no tiene más que teclear SYS 828 desde el BASIC o G$033C desde el monitor, y todo arreglado.

PROGRAMA A. RUTINA LIST

```
033C A2 20      LDX #$20
033E A9 A0      LDA #$A0
0340 A0 00      LDY #$00
0342 84 22      STY $22
0344 85 23      STA $23
0346 B1 22      LDA ($22),Y
0348 91 22      STA ($22),Y
034A C8         INY
034B D0 F9      BNE $0346
034D E6 23      INC $23
034F CA         DEX
0350 D0 F4      BNE $0346
0352 A9 60      LDA #$60
0354 8D 14 A7   STA $A714
0357 A9 EA      LDA #$EA
0359 8D BB A6   STA $A6BB
035C 8D BC A6   STA $A6BC
035F A9 6D      LDA #$6D
0361 8D 42 A0   STA $A042
0364 A9 03      LDA #$03
0366 8D 43 A0   STA $A043
0369 A9 36      LDA #$36
036B 85 01      STA $01
036D 60         RTS
036E A5 7A      LDA $7A
0370 48         PHA
0371 A5 7B      LDA $7B
0373 48         PHA
0374 20 73 00   JSR $0079
0377 20 9C A6   JSR $A69C
037A 20 D7 AA   JSR $AAD7
037D 68         PLA
```



Ahora ya puede introducir en su programa BASIC la sentencia LIST o LIST X. Después de hacer el listado, el programa continuará normalmente.

Modificación de GOTO, GOSUB y RESTORE

Es lo que se suele llamar bifurcación o restauración por cálculo. Muchas veces nos sería muy útil poder introducir en BASIC los siguientes comandos:

GOTO A−3 B
GOSUB C+5
RESTORE 10

Esto provocaría inmediatamente un "Syntax error". Sin embargo, se pueden modificar las rutinas del interpretador de tal forma que esos comandos sean visibles.

De esta manera podemos escribir una línea BASIC que bifurque a líneas diferentes según el valor calculado.

También podemos hacer que se lea un determinado DATA, sin necesidad de leer todos los anteriores sin más que hacer RESTORE N, siendo N el número de dato a partir del cual queremos hacer la lectura.

A continuación se presenta el listado de la rutina.

```
C000 A2 A0      LDX #$A0
C002 A0 00      LDY #$00
C004 86 FF      STX $FF
C006 84 FE      STY $FE
C008 A2 20      LDX #$20
C00A B1 FE      LDA ($FE),Y
C00C 91 FE      STA ($FE),Y
C00E C8         INY
C00F D0 F9      BNE $C00A
```

```
C011 E6 FF      INC $FF
C013 CA         DEX
C014 D0 F4      BNE $C00A
C016 60         RTS
C017 20 FD AE   JSR $AEFD
C01A 20 9E B7   JSR $B79E
C01D 86 FE      STX $FE
C01F 20 FD AE   JSR $AEFD
C022 20 9E B7   JSR $B79E
C023 A4 FE      LDY $FE
C027 18         CLC
C028 20 F0 FF   JSR $FFF0
C02B 20 FD AE   JSR $AEFD
C02E 4C A0 AA   JMP $AAA0
C031 20 8A AD   JSR $AD8A
C034 4C F7 B7   JMP $B7F7
C037 20 00 C0   JSR $C000
C03A A9 C0      LDA #$C0
C03C A2 31      LDX #$31
C03E 8D A2 A8   STA $A8A2
C041 8E A1 A8   STX $A8A1
C044 A9 36      LDA #$36
C046 85 01      STA $01
C048 60         RTS
C049 EA         NOP
C04A F0 15      BEQ $C061
C04C 20 31 C0   JSR $C031
C04F 20 13 A6   JSR $A613
C052 38         SEC
C053 A5 5F      LDA $5F
C055 E9 01      SBC #$01
C057 A4 60      LDY $60
C059 B0 01      BCS $C05C
C05B 88         DEY
C05C 85 41      STA $41
C05E 84 42      STY $42
C060 60         RTS
C061 4C 1D A8   JMP $A81D
C064 20 37 C0   JSR $C037
C067 A9 C0      LDA #$C0
C069 A2 49      LDX #$49
C06B 8D 25 A0   STA $A025
C06E 8E 24 A0   STX $A024
C071 60         RTS
```

La rutina se compone de cuatro partes:

1.ª) C000-C016. Copia la ROM a la RAM que hay debajo de ella desde la posición $A000 (40960) hasta la $BFFF (49151).

2.ª) C017-C830. Rutina PRINT AT. Sitúa el cursor y escribe en la posición indicada. La sintaxis es: SYS 49175, columna, fila, "texto". Puede introducirse sola, sin ninguna de las otras tres partes.

3.ª) C031-C048. Rutina GOTO N y GOSUB N. Lo único que hay que hacer es un SYS 49207 al principio del programa y después, cuando la vayamos a utilizar, teclear dentro del programa BASIC GOTO N o GOSUB N. Debe estar en memoria la parte 1.ª

4.ª) C049-C071. Rutina RESTORE N. Para utilizarlo deben estar en memoria las partes 1.ª y 3.ª Teclee SYS 49252 en una línea de programa BASIC antes de utilizar el comando RESTORE N, y todo listo.

Rutinas SAVE y LOAD

Para terminar, daremos un esquema de cómo utilizar estas dos rutinas del sistema operativo en zonas de memoria que no correspondan al área BASIC.

La rutina LOAD comprende tres rutinas:

```
  LOAD  $FFD8(65496)
SETFLS  $FFBA(65466)
SETNAM  $FFBD(65469)
```

Para cargar/verificar debe hacerse lo siguiente: Cargar el ACU con el número de fichero lógico (0,1), en X el número de dispositivo (0 = teclado, 1 = cinta, 2 = RS232C, 3 = pantalla, 4 = impresora, 8 = disco) y llamar a SETFLS con JSR $FFBA.

A continuación cargue en el ACU la longitud del nombre de fichero y en X e Y el low-byte y high-byte de la dirección de inicio. Si no hace falta el nombre, en el ACU se debe haber introducido un $00. Llame ahora a la rutina SET NAM con JSR $FFBD.

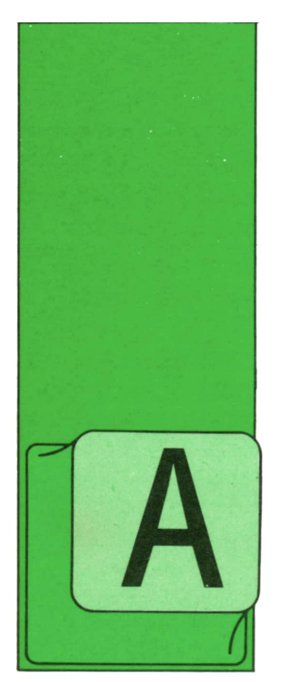
Por último, cargue el ACU con (1 = verify, 0 = Load), y llame a LOAD con JSR $FFD5.

Para grabar, siga los mismos pasos de llamada a SETLFS y SETNAM. Después cargue dos posiciones consecutivas de la página cero (ejemplo: $FB y $FC) con el Low-byte y High-byte de la dirección de comienzo, y el ACU con $FB. Cargue en X e Y el Low-byte + 1 y High-byte de la dirección final a salvar. Ahora llame a la rutina SAVE con JSR $FFD8.

Bien, ya puede empezar a experimentar con todo lo que tiene, ya que es el único medio de aprender y llegar a familiarizarse con los comandos y rutinas usados en lenguaje máquina. Con un poco de tiempo podrá programar sus propias rutinas originales.

PROGRAMAS

Programa: PAC-MAN para AMSTRAD



PARECE a continuación el programa PAC-MAN, que ya ha aparecido en versión para otros ordenadores.

Como todos sabemos, el objetivo del juego es comerse el mayor número posible de puntos antes de que los fantasmas, que pueblan el laberinto que aparece en nuestra pantalla, nos asesinen. Para ello debemos controlar a nuestro quesito (PAC-MAN) con las siguientes teclas:

O — Movernos hacia la izquierda.
P — Movernos hacia la derecha.
Q — Movernos hacia arriba.
A — Movernos hacia abajo.

En el laberinto, aparte de los puntos, podemos ver cuatro asteriscos que nos darán cincuenta puntos si nos los comemos. En la versión original de este juego, cada vez que nos comíamos uno de estos puntos, los fantasmas que nos perseguían cambiaban de forma y podíamos comérnoslos. A cambio, cada vez que nos comamos un asterisco aumentará nuestra velocidad durante un breve espacio de tiempo. Tenemos que aprovechar dicha oportunidad para huir de los fantasmas y para comernos más puntos.

El número de fantasmas con el que nos tenemos que enfrentar se reduce a dos. Esto ha sido necesario para que el programa no se volviese demasiado lento.

```
PROGRAMA: PAC-MAN PARA AMSTRAD
==============================

1000 REM *************************
1010 REM *                       *
1020 REM *    P A C  -  M A N    *
1030 REM *                       *
1040 REM *************************
1050 REM
1060 REM *************************
1070 REM *                       *
1080 REM *       A U T O R       *
1090 REM *      -----------      *
1100 REM *                       *
1110 REM * Fco. Morales Guerrero *
1120 REM *                       *
1130 REM *************************
1140 REM
1150 REM **************************
1160 REM *                        *
1170 REM * (c) Ed. Siglo Cultural *
1180 REM * (c) 1987.              *
1190 REM *                        *
1200 REM **************************
1210 REM
```

```
1220 REM ********************************
1230 REM * INICIALIZACION DEL PROGRAMA *
1240 REM ********************************
1250 REM
1260 DIM N(9),N$(10),P(10)
1270 DIM S(25,41),M$(2,3),I(2,8)
1280 LET M$(1,0)="."
1290 LET M$(1,1)="*"
1300 LET M$(1,3)=" "
1310 LET M$(2,0)="."
1320 LET M$(2,1)="*"
1330 LET M$(2,3)=" "
1340 LET PA=1
1350 MODE 1
1360 PEN 3
1370 CLS
1380 PRINT TAB(15);"COME - COCOS"
1390 PRINT TAB(14);"=============="
1400 PEN 1
1410 LOCATE 10,14:PRINT "CARGANDO DATAS"
1420 PEN 2
1430 LOCATE 12,12:PRINT "ESPERA UN MOMENTO."
1440 REM
1450 REM *********************************
1460 REM * DEFINICION DE LOS PERSONAJES *
1470 REM *********************************
1480 REM
1490 SYMBOL 255,60,126,255,255,255,255,126,60 : REM PAC-MAN CERRADO
1500 SYMBOL 254,60,126,255,15,15,255,126,60   : REM PAC-MAN IZQUIERDA
1510 SYMBOL 253,60,126,255,240,240,255,126,60 : REM PAC-MAN DERECHA
1520 SYMBOL 252,36,102,231,231,255,255,126,60 : REM PAC-MAN ARRIBA
1530 SYMBOL 251,60,126,255,255,231,231,102,36 : REM PAC-MAN ABAJO
1540 SYMBOL 250,60,126,219,219,126,60,36,66   : REM FANTASMA-1 POSICION A
1550 SYMBOL 249,60,126,219,219,126,60,36,36   : REM FANTASMA-1 POSICION B
1560 SYMBOL 248,126,255,189,219,126,36,66,129 : REM FANTASMA-2 POSICION A
1570 SYMBOL 247,126,255,189,219,126,36,36,36  : REM FANTASMA-2 POSICION B
1580 REM
1590 REM *****************************
1600 REM * LECTURA DE LOS CARACTERES *
1610 REM *****************************
1620 REM
1630 FOR I=0 TO 9
1640    READ N(I)
1650 NEXT I
1660 REM
1670 REM **************************
1680 REM * LECTURA DE LOS RECORDS *
1690 REM **************************
1700 REM
1710 FOR I=1 TO 10
1720    READ N$(I),P(I)
1730 NEXT I
1740 REM
1750 REM ******************************
1760 REM * LECTURA DE LAS DIRECCIONES *
1770 REM ******************************
1780 REM
1790 FOR I=1 TO 2
1800    FOR J=1 TO 8
1810       READ I(I,J)
1820    NEXT J
1830 NEXT I
1840 REM
1850 REM **********************
1860 REM * PROGRAMA PRINCIPAL *
1870 REM **********************
1880 REM
1890 CLS
```

```
1900 GOSUB 5590:REM PRESENTACION
1910 GOSUB 4730:REM TABLA DE RECORDS
1920 GOSUB 5340:REM INSTRUCCIONES
1930 GOSUB 5490:REM PULSA UNA TECLA
1940 GOSUB 4200:REM DIBUJA LA PANTALLA
1950 PEN 2
1960 GOSUB 2020 :REM LEE EL TECLADO
1970 GOSUB 2170 :REM MUEVE EL COME-COME
1980 GOSUB 2530 :REM ENEMIGO NUMERO 1
1990 GOSUB 2760 :REM ENEMIGO NUMERO 2
2000 IF PT<=0 THEN GOSUB 3730:LET PA=PA+1:GOTO 1940
2010 GOTO 1960
2020 REM
2030 REM **********************
2040 REM * CONTROL DEL TECLADO *
2050 REM **********************
2060 REM
2070 LET I$=INKEY$:IF I$="" THEN LET I$=Z$
2080 LET Z$=I$
2090 LET II=INSTR("OoPpQqAa",I$)
2100 IF II=0 THEN RETURN
2110 LET X1=X0:LET Y1=Y0
2120 LET X0=X0+I(1,II)+I(1,II)*(S(Y0,X0+I(1,II))=0)
2130 LET Y0=Y0+I(2,II)+I(2,II)*(S(Y0+I(2,II),X0)=0)
2140 IF X0=41 THEN LET X0=1
2150 IF X0=0 THEN LET X0=40
2160 RETURN
2170 REM
2180 REM ***************************
2190 REM * MOVIMIENTO DEL COME-COME *
2200 REM ***************************
2210 REM
2220 IF BB>0 THEN LET BB=BB+1:IF BB=60-(10*PA) THEN LET BB=0
2230 IF X1=X0 AND Y1=Y0 THEN LET Z$="Z":RETURN
2240 IF S(Y0,X0)=-1 THEN LET PT=PT-1:LET S(Y0,X0)=2:LET PU=PU+10:LOCATE 13,2:PRI
NT USING "######";PU
2250 IF S(Y0,X0)=1 THEN LET PT=PT-1:LET S(Y0,X0)=2:LET PU=PU+50:LOCATE 13,2:PRIN
T USING "######";PU:LET BB=1
2260 IF PU>=HI THEN LOCATE 32,2:PRINT USING "######";PU:LET HI=PU
2270 LOCATE X1,Y1:PRINT " "
2280 LET S(Y1,X1)=2
2290 IF SW=1 THEN LOCATE X0,Y0:PRINT CHR$(255):LET SW=0:LET S(Y0,X0)=99:RETURN
2300 ON II GOSUB 2330,2330,2380,2380,2430,2430,2480,2480
2310 LET SW=1:LET S(Y0,X0)=99
2320 RETURN
2330 REM
2340 REM *** MOVIMIENTO A LA IZQUIERDA ***
2350 REM
2360 LOCATE X0,Y0:PRINT CHR$(254)
2370 RETURN
2380 REM
2390 REM *** MOVIMIENTO A LA DERECHA ***
2400 REM
2410 LOCATE X0,Y0:PRINT CHR$(253)
2420 RETURN
2430 REM
2440 REM *** MOVIMIENTO A ARRIBA ***
2450 REM
2460 LOCATE X0,Y0:PRINT CHR$(252)
2470 RETURN
2480 REM
2490 REM *** MOVIMIENTO A ABAJO ***
2500 REM
2510 LOCATE X0,Y0:PRINT CHR$(251)
2520 RETURN
2530 REM
2540 REM *****************************
```

```
2550 REM * MOVIMIENTO DEL FANTASMA-1 *
2560 REM ******************************
2570 REM
2580 REM *** CALCULO DE LA DIRECCION ***
2590 REM
2600 LET SO=0:LET MM=MM+1
2610 IF BB>0 AND SD1=1 THEN LET SD1=0:RETURN
2620 LET SD1=1
2630 LET X3=X2:LET Y3=Y2
2640 LET SX=SGN(X1-X2)
2650 LET SY=SGN(Y1-Y2)
2660 LET X2=X2+SX+SX*(S(Y2,X2+SX)=0)
2670 LET Y2=Y2+SY+SY*(S(Y2+SY,X2)=0)
2680 REM
2690 REM *** MOVIMIENTO ***
2700 REM
2710 LOCATE X3,Y3:PRINT M$(1,M1+1)
2720 LET M1=S(Y2,X2)
2730 IF M1=99 THEN GOTO 3010
2740 IF SE1=1 THEN LET SE1=0:LOCATE X2,Y2:PRINT CHR$(250):RETURN
2750 LET SE1=1:LOCATE X2,Y2:PRINT CHR$(249):RETURN
2760 REM
2770 REM ******************************
2780 REM * MOVIMIENTO DEL FANTASMA-2 *
2790 REM ******************************
2800 REM
2810 REM *** CALCULO DE LA DIRECCION ***
2820 REM
2830 LET SO=0
2840 IF BB>0 AND LET SD2=1 THEN LET SD2=0:RETURN
2850 LET SD2=1
2860 IF X3=X4 AND Y3=Y4 THEN LET MM=9
2870 IF MM<15 THEN LOCATE X4,Y4:PRINT CHR$(248):RETURN
2880 LET X5=X4:LET Y5=Y4
2890 LET SX=SGN(X1-X4)
2900 LET SY=SGN(Y1-Y4)
2910 LET X4=X4+SX+SX*(S(Y4,X4+SX)=0)
2920 LET Y4=Y4+SY+SY*(S(Y4+SY,X4)=0)
2930 REM
2940 REM *** MOVIMIENTO ***
2950 REM
2960 LOCATE X5,Y5:PRINT M$(2,M2+1)
2970 LET M2=S(Y4,X4)
2980 IF M2=99 THEN GOTO 3010
2990 IF SE2=1 THEN LET SE2=0:LOCATE X4,Y4:PRINT CHR$(247):RETURN
3000 LET SE2=1:LOCATE X4,Y4:PRINT CHR$(248):RETURN
3010 REM
3020 REM ***********************
3030 REM * PERDIDA DE UNA VIDA *
3040 REM ***********************
3050 REM
3060 LOCATE X0,Y0:PRINT " "
3070 FOR I=1 TO 10
3080    LOCATE X0,Y0:PRINT CHR$(255)
3090    FOR K=1 TO 100
3100    NEXT K
3110    LOCATE X0,Y0:PRINT " "
3120    FOR K=1 TO 100
3130    NEXT K
3140 NEXT I
3150 LOCATE Y0,X0:PRINT " "
3160 LOCATE Y2,X2:PRINT " "
3170 LOCATE Y4,X4:PRINT " "
3180 LET NV=NV-1
3190 IF NV=0 THEN GOTO 3440
3200 LET Y0=16
3210 LET X0=20
```

```
3220 LET Y1=Y0
3230 LET X1=X0
3240 LET Z$="Z
3250 LET BB=0
3260 LET X2=19
3270 LET Y2=11
3280 LET X3=X2
3290 LET Y3=Y2
3300 LET X4=21
3310 LET Y4=11
3320 LET X5=X4
3330 LET Y5=Y4
3340 LET M1=S(Y2,X2)
3350 LET M2=S(Y4,X4)
3360 LOCATE 10,24:PRINT "    ";
3370 FOR I=1 TO NV-1
3380    LOCATE 9+I,24:PRINT CHR$(253)
3390 NEXT I
3400 LOCATE X0,Y0:PRINT CHR$(243)
3410 GOSUB 3610
3420 RETURN
3430 RETURN
3440 REM
3450 REM *************
3460 REM * GAME OVER *
3470 REM *************
3480 REM
3490 FOR I=1 TO 5
3500    FOR J=1 TO 3
3510       PEN J
3520       LOCATE 18,13:PRINT " GAME "
3530       LOCATE 18,14:PRINT " OVER "
3540       FOR K=1 TO 140:NEXT K
3550    NEXT J
3560    LOCATE 18,13:PRINT "      "
3570    LOCATE 18,14:PRINT "      "
3580    FOR K=1 TO 140:NEXT K
3590 NEXT I
3600 GOTO 1910
3610 REM
3620 REM ***************
3630 REM * ESTAS LISTO *
3640 REM ***************
3650 REM
3660 LOCATE 18,13:PRINT "(ESTAS"
3670 LOCATE 18,14:PRINT "LISTO?"
3680 FOR J=1 TO 1000
3690 NEXT J
3700 LOCATE 18,13:PRINT "      "
3710 LOCATE 18,14:PRINT "      "
3720 RETURN
3730 REM
3740 REM ***************************
3750 REM * TERMINO DE UNA PANTALLA *
3760 REM ***************************
3770 REM
3780 FOR I=1 TO 10
3790    BORDER 12
3800    FOR J=1 TO 100
3810    NEXT J
3820    BORDER 0
3830    FOR J=1 TO 100
3840    NEXT J
3850 NEXT I
3860 CLS
3870 PEN 3
3880 PRINT "MUY BIEN ...."
```

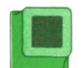
```
3890 PRINT
3900 PRINT TAB(10);"TERMINASTE LA PANTALLA"
3910 FOR I=2 TO 40
3920    LOCATE I,10
3930    PRINT "."
3940 NEXT I
3950 FOR I=1 TO 10
3960    LOCATE 1,10
3970    PRINT CHR$(253)
3980    FOR J=1 TO 100
3990    NEXT J
4000    LOCATE 1,10
4010    PRINT " "
4020    FOR J=1 TO 100
4030    NEXT J
4040 NEXT I
4050 LOCATE 1,10
4060 PRINT CHR$(253)
4070 FOR I=2 TO 40
4080    LOCATE I,10:PRINT " ";CHR$(253)
4090    FOR J=1 TO 100:NEXT J
4100    LOCATE I,10:PRINT " ";CHR$(255)
4110    FOR J=1 TO 100:NEXT J
4120 NEXT I
4130 PRINT
4140 PRINT " HAS CONSEGUIDO ..."
4150 PRINT
4160 PRINT PA*600;"PUNTOS DE BONUS"
4170 LET PU=PU+PA*600
4180 GOSUB 5490
4190 RETURN
4200 REM
4210 REM *************************
4220 REM * DIBUJO DE LA PANTALLA *
4230 REM *************************
4240 REM
4250 CLS
4260 RESTORE 6390
4270 FOR I=1 TO 25
4280    IF I=25 THEN LOCATE 1,25
4290    READ A$
4300    FOR J=1 TO LEN(A$)
4310       LET S(I,J)=0
4320       LET B$=MID$(A$,J,1)
4330       IF B$="." OR B$="*" THEN PEN 2:PRINT B$;:LET S(I,J)=(B$=".")-(B$="*")
:PEN 3:GOTO 4390
4340       IF B$>"9" THEN GOTO 4360
4350       PRINT CHR$(N(VAL(B$)));:GOTO 4390
4360       IF B$="A" THEN PRINT CHR$(155);
4370       IF B$="B" THEN PRINT CHR$(158);
4380       IF B$="C" THEN PRINT CHR$(140);
4390    NEXT J
4400 NEXT I
4410 PEN 3
4420 LOCATE 4,2:PRINT USING "PUNTOS = ######";PU
4430 LOCATE 27,2:PRINT USING "HI = ######";HI
4440 LOCATE 2,24:PRINT "VIDAS = ";
4450 LOCATE 28,24:PRINT USING "PANTALLA = #";PA;
4460 FOR I=1 TO NV-1
4470    LOCATE 9+I,24:PRINT CHR$(253)
4480 NEXT I
4490 LET Y0=16
4500 LET X0=20
4510 LET Y1=Y0
4520 LET X1=X0
4530 LET Z$="Z"
4540 LET S(13,41)=2
```

```
4550 LET S(13,0)=2
4560 LET BB=0
4570 LET MM=0
4580 LET M2=-1
4590 LET PT=287
4600 LET X2=19
4610 LET Y2=11
4620 LET X3=X2
4630 LET Y3=Y2
4640 LET X4=21
4650 LET Y4=11
4660 LET X5=X4
4670 LET Y5=Y4
4680 LET M1=-1
4690 LOCATE X0,Y0
4700 PRINT CHR$(253)
4710 GOSUB 3610
4720 RETURN
4730 REM
4740 REM ********************
4750 REM * TABLA DE RECORDS *
4760 REM ********************
4770 REM
4780 CLS
4790 PEN 3
4800 PRINT TAB(13);"TABLA DE RECORDS"
4810 PRINT TAB(12);"=================="
4820 PEN 1
4830 PRINT TAB(6);CHR$(150);STRING$(28,154);CHR$(156)
4840 PRINT TAB(6);CHR$(149);SPC(28);CHR$(149)
4850 PRINT TAB(6);CHR$(151);STRING$(28,154);CHR$(151)
4860 FOR I=1 TO 9
4870    PRINT TAB(6);CHR$(149);SPC(28);CHR$(149)
4880 NEXT I
4890 PRINT TAB(6);CHR$(147);STRING$(28,154);CHR$(153)
4900 NP=0
4910 FOR I=1 TO 10
4920    IF PU>P(I) THEN LET NP=1
4930 NEXT I
4940 IF NP=1 THEN LET P(10)=PU:LET N$(10)=CHR$(255)
4950 FOR I=1 TO 10
4960    FOR J=1 TO I
4970       IF P(I)>P(J) THEN LET A$=N$(I):LET A=P(I):LET N$(I)=N$(J):LET P(I)=P(
J):LET N$(J)=A$:LET P(J)=A
4980    NEXT J
4990 NEXT I
5000 IF NP=0 THEN GOTO 5050
5010 FOR I=1 TO 10
5020    IF N$(I)=CHR$(255) THEN LET N$(I)=STRING$(15,"-"):LET NP=I
5030 NEXT I
5040 PEN 2
5050 LOCATE 7,4:PRINT N$(1);TAB(23);"... ";USING "######";P(1)
5060 FOR I=2 TO 10
5070    LOCATE 7,4+I
5080    PRINT N$(I);TAB(23);"... ";USING "######";P(I)
5090 NEXT I
5100 IF NP=0 THEN GOTO 5290
5110 PEN 3
5120 LOCATE 15,16:PRINT "FELICIDADES."
5130 PRINT
5140 PRINT TAB(3);"TU PUNTUACION ESTA ENTRE LAS MEJORES"
5150 PRINT
5160 PRINT TAB(12);"ESCRIBE TU NOMBRE."
5170 LET N$(NP)="":LET X=7:LET LO=0:LET D$=""
5180 IF NP=1 THEN LET Y=4 ELSE LET Y=4+NP
5190 LOCATE X,Y
5200 PRINT "_";CHR$(29);
```

```
5210 LET A$=INKEY$:IF A$="" THEN GOTO 5210
5220 IF A$=CHR$(8) AND LO>0 THEN LET X=X-1:LET LO=LO-1:LET D$=LEFT$(D$,LO):PRINT
 CHR$(29);"_-";CHR$(29);CHR$(29);
5230 IF A$=" " THEN GOTO 5260
5240 IF A$=CHR$(13) THEN GOTO 5270
5250 IF A$<"0" OR A$>"z" THEN GOTO 5210
5260 PRINT A$;"_";CHR$(29);:LET LO=LO+1:LET D$=D$+A$:IF LO<>15 THEN GOTO 5210
5270 PRINT SPC(16-LO)
5280 LET N$(NP)=D$
5290 LET HI=P(1)
5300 LOCATE 1,16:PRINT SPACE$(240)
5310 LET PU=0:LET NV=3
5320 IF NP<>0 THEN LET NP=0:GOSUB 5590:GOTO 4730
5330 RETURN
5340 REM
5350 REM ****************
5360 REM * INSTRUCCIONES *
5370 REM ****************
5380 REM
5390 PEN 3
5400 LOCATE 4,17
5410 PRINT "COMETE TODOS LOS PUNTOS QUE PUEDAS"
5420 PRINT
5430 PRINT "    ^"
5440 PRINT "    Q          USA ESTAS TECLAS"
5450 PRINT "< O   P >"
5460 PRINT "    A        PARA MOVER EL QUESITO."
5470 PRINT "    v"
5480 RETURN
5490 REM
5500 REM ******************
5510 REM * PULSA UNA TECLA *
5520 REM ******************
5530 REM
5540 LOCATE 13,25
5550 PRINT "PULSA UNA TECLA"
5560 LET A$=INKEY$
5570 IF A$="" THEN GOTO 5560
5580 RETURN
5590 REM
5600 REM ***************
5610 REM * PRESENTACION *
5620 REM ***************
5630 REM
5640 CLS
5650 PEN 3
5660 RESTORE 6690
5670 FOR I=2 TO 16
5680    READ A$
5690    FOR J=1 TO LEN(A$)
5700       IF MID$(A$,J,1)="0" THEN GOTO 5730
5710       LOCATE 7+J,I+1)
5720       PRINT CHR$(253)
5730    NEXT J
5740 NEXT I
5750 LOCATE 6,1:PRINT "(c) Ed. Siglo Cultural, 1987"
5760 PEN 2
5770 FOR I=1 TO 14
5780    LOCATE I,20:PRINT "."
5790    FOR J=1 TO 100:NEXT J
5800 NEXT I
5810 PEN 3
5820 LOCATE 17,20:PRINT "= 10 PUNTOS"
5830 PEN 2
5840 FOR I=1 TO 13
5850    LOCATE I,22:PRINT " *"
5860    FOR J=1 TO 100:NEXT J
```

```
5870 NEXT I
5880 PEN 3
5890 LOCATE 17,22:PRINT "= 50 PUNTOS"
5900 FOR J=1 TO 1000:NEXT J
5910 FOR I=1 TO 15
5920    LOCATE I,20:PRINT CHR$(253)
5930    FOR J=1 TO 100:NEXT J
5940    LOCATE I,20:PRINT CHR$(255)
5950    FOR J=1 TO 100:NEXT J
5960    LOCATE I,20:PRINT "  "
5970 NEXT I
5980 LOCATE 15,20:PRINT CHR$(253)
5990 LOCATE 17,20:PRINT "= TU MISMO "
6000 FOR I=1 TO 15
6010    LOCATE I,22:PRINT CHR$(250)
6020    FOR J=1 TO 100:NEXT J
6030    LOCATE I,22:PRINT CHR$(249)
6040    FOR J=1 TO 100:NEXT J
6050    LOCATE I,22:PRINT "  "
6060 NEXT I
6070 LOCATE 15,22:PRINT CHR$(250)
6080 LOCATE 17,22:PRINT "= TU ENEMIGO"
6090 FOR I=1 TO 1000
6100 NEXT I
6110 RETURN
6120 REM
6130 REM ****************************
6140 REM * DATAS CON LOS CARACTERES *
6150 REM ****************************
6160 REM
6170 DATA 32,150,156,147,153,154,149,159,151,157
6180 REM
6190 REM *************************
6200 REM * DATAS CON LOS RECORDS *
6210 REM *************************
6220 REM
6230 DATA "ANTONIO",34990
6240 DATA "EL COCO",31720
6250 DATA "EL AUTOR",24030
6260 DATA "INDIANA",23300
6270 DATA "JOSELITO",11210
6280 DATA "SUSYN",11120
6290 DATA "RAMIRO",8430
6300 DATA "DONALD",7540
6310 DATA "SHE",3400
6320 DATA "HE",3210
6330 REM
6340 REM *****************************
6350 REM * DATAS CON LAS DIRECCIONES *
6360 REM *****************************
6370 REM
6380 DATA -1,-1,1,1,0,0,0,0,0,0,0,0,-1,-1,1,1
6390 REM
6400 REM ****************************************
6410 REM * DATAS CON LA DEFINICION DE LA PANTALLA *
6420 REM ****************************************
6430 REM
6440 DATA "0015555555555555552000000015555555555200"
6450 DATA "0060000000000006000000060000000000000600"
6460 DATA "15A555555555555B5A5555B5A555555555555A52"
6470 DATA "6*..............35555554..............*6"
6480 DATA "6.15552.1555552..........1555552:15552.6"
6490 DATA "6.60006.600000352.1552.154000006.60006.6"
6500 DATA "6.35554.355555554.6006.355555554.35554.6"
6510 DATA "6.................6006.................6"
6520 DATA "8555552.155552.1554003552.155552.1555559"
6530 DATA "6155526.601554.3555555554.355206.6155526"
```

```
6540 DATA "6355546.606..................606.6355546"
6550 DATA "3555554.354.152.1CCCCCC2.152.354.3555554"
6560 DATA "............606.60000006.606..........."
6570 DATA "15555B2.152.606.60000006.606.152.1B55552"
6580 DATA "6....34.354.354.35555554.354.354.34....6"
6590 DATA "6.12...............................12.6"
6600 DATA "6.63552.1555555555555555555555552.15546.6"
6610 DATA "6.35526.6000155555555555555520006.61554.6"
6620 DATA "6....34.35554..............35554.34....6"
6630 DATA "6.12..........15B555555B52..........12.6"
6640 DATA "6.34.15555552.354......354.15555552.34.6"
6650 DATA "6*...35555206.....1552.....60155554...*6"
6660 DATA "8555555555A5755555A55A5555BA5A555555559"
6670 DATA "6000000000006000000000000006000000000006"
6680 DATA "3555555555554000000000000003555555555554"
6690 REM
6700 REM ********************************
6710 REM * DATAS CON LA PALABRA PAC-MAN *
6720 REM ********************************
6730 REM
6740 DATA "1111110000110000011110"
6750 DATA "0110011001111000110011"
6760 DATA "0110011011001101100000"
6770 DATA "0111100011001101100000"
6780 DATA "0110000011111101100000"
6790 DATA "0110000011001100110011"
6800 DATA "1111000011001100011110"
6810 DATA ""
6820 DATA "11000110000110001100011"
6830 DATA "11101110001111001110011"
6840 DATA "11111110011001101111011"
6850 DATA "11111110011001101101111"
6860 DATA "11010110011111101100111"
6870 DATA "11000110011001101100011"
6880 DATA "11000110011001101100011"
```



Programa: Tipo parabólico

En los programas educativos, y, dentro de éstos, en los programas de física, uno de los temas más apasionantes y divertidos es el del lanzamiento de proyectiles o tiro oblicuo.

Con este programa, realizado para el SPECTRUM, podremos resolver problemas de tiro oblicuo en cualquiera de los planetas que existen en nuestro sistema solar.

Resumiendo, el programa nos permite hacer lo siguiente:

— Plantear un problema real y resolverlo numéricamente.

— Cambiar el planeta donde se realiza el lanzamiento de los proyectiles.

— Ver la representación gráfica de dicho movimiento.

— Ver en qué unidades medimos las diferentes características de cada lanzamiento.

— Ver las ecuaciones que utilizamos.

```
PROGRAMA: TIRO PARABOLICO
-------------------------

   1 REM **********************
   2 REM * FISICA:TIRO OBLICUO *
   3 REM **********************
   4 REM *JOSE M. GARCIA LUENGO*
   5 REM **********************
   6 LET X$="(VO^2*SIN (2*ANG))/G": LET Y$="VO^2*(SIN ANG)^2/(2*G)": LET NAD=1:
LET G=9.8
   7 CLS : LET NOP=6: LET EL=1
   8 PRINT AT 0,0; INVERSE 1;"* * T I R O    O B L I C U O * *"
   9 PRINT
  10 PRINT  BRIGHT 1;"MENU PRINCIPAL:"
  20 PRINT : PRINT  FLASH 1;"1";
  30 PRINT ". REPRESENTACION."
  40 PRINT "2. PROBLEMAS."
  50 PRINT "3. GRAVEDADES DEL SISTEMA SOLAR."
  60 PRINT "4. UNIDADES."
  70 PRINT "5. ECUACIONES GENERALES."
  80 PRINT "6. GOTO OPCION."
  90 GO SUB 100: IF VAL K$=NOP THEN  GO TO 1000*EL-1
  95 GO TO 90
 100 LET K$=INKEY$
 110 IF K$="" OR K$>STR$ NOP OR K$<STR$ 1 THEN  GO TO 100
 120 BEEP .01,50
 130 IF VAL K$=NOP THEN  RETURN
 140 PRINT AT EL+3,0; FLASH 0;EL
 150 LET EL=VAL K$
 160 PRINT AT EL+3,0; FLASH 1;EL
 170 RETURN
 998 REM
 999 REM ******************
1000 REM * REPRESENTACION *
1001 REM ******************
1002 REM
1010 CLS
1020 IF NAD=1 THEN  PRINT AT 10,0; FLASH 1;"NO HAY DATOS EN MEMORIA.........": B
EEP .1,0: PAUSE 0: GO TO 6
1030 PLOT 7,32
1040 DRAW 0,143
1050 DRAW -2,-7
1060 DRAW 4,0
1070 DRAW -2,7
1080 PRINT AT 0,2;"y"
1090 PLOT 0,47
1100 DRAW 254,0
1110 DRAW -7,2
1120 DRAW 0,-4
1130 DRAW 7,2
1140 PRINT AT 14,31;"x"
1150 PLOT 7,47
1160 DRAW 216,0,-PI/2
1170 PLOT 112,48
1180 DRAW 0,48
1190 PRINT AT 9,11;"H(max)"
1200 PLOT 7,47
1210 DRAW 40,40
1220 DRAW -7,-3
1230 DRAW 4,-4
1240 DRAW 3,7
1250 PRINT AT 10,4;"Vo"
1260 PRINT AT 18,0;"X=T=0"
1270 PLOT 19,47
1280 DRAW -4,6,PI/4
1290 PRINT AT 15,3;"@"
1300 PRINT AT 16,0; OVER 1;"o"
```

```
1310 PLOT 223,50
1320 DRAW 0,-10
1330 PRINT AT 17,27;"X"
1340 PRINT AT 18,27;"T"
1350 PRINT AT 0,5;"x= ";X
1360 PRINT AT 1,5;"Vo= ";VO
1370 PRINT AT 2,5;"t= ";T
1380 PRINT AT 3,5;"@= ";ANG
1390 PRINT AT 4,5;"g= ";G
1400 PRINT AT 5,5;"H(max)= ";Y
1420 PRINT #1; FLASH 1;"PULSA UNA TECLA ."
1430 PAUSE 0
1440 GO TO 7
1998 REM
1999 REM *************
2000 REM * PROBLEMAS *
2001 REM *************
2002 REM
2010 CLS : PRINT
2020 PRINT  INVERSE 1;"** ELIGE LA PRIMERA INCOGNITA **"
2030 GO SUB 2800
2040 LET NOP=6: LET EL=1
2050 GO SUB 100: IF VAL K$=NOP THEN  GO TO 2070
2060 GO TO 2050
2070 LET E1=EL
2080 CLS : PRINT
2090 PRINT  INVERSE 1;"* *ELIGE LA SEGUNDA INCOGNITA* *"
2100 GO SUB 2800
2110 PRINT AT 20,0; BRIGHT 1;"YA HAS ELEGIDO LA NUMERO ";E1
2120 LET NOP=6: LET EL=1
2130 GO SUB 100: IF VAL K$=NOP AND EL<>E1 THEN  GO TO 2150
2140 GO TO 2130
2150 LET E2=EL
2160 CLS
2170 IF E1=1 THEN  LET FO=(1 AND E2=2)+(2 AND E2=3)+(3 AND E2=4)+(4 AND E2=5)
2180 IF E1=2 THEN  LET FO=(1 AND E2=1)+(5 AND E2=3)+(6 AND E2=4)+(7 AND E2=5)
2190 IF E1=3 THEN  LET FO=(2 AND E2=1)+(5 AND E2=2)+(8 AND E2=4)+(9 AND E2=5)
2200 IF E1=4 THEN  LET FO=(3 AND E2=1)+(6 AND E2=2)+(8 AND E2=3)+(10 AND E2=5)
2210 IF E1=5 THEN  LET FO=(4 AND E2=1)+(7 AND E2=2)+(9 AND E2=3)+(10 AND E2=4)
2220 LET NAD=0: CLS : GO SUB 5100+(FO-1)*100
2230 CLS
2240 PRINT  INVERSE 1;"* * R E S U L T A D O S * *"
2250 PRINT : PRINT
2260 PRINT "x= ";X: PRINT
2270 PRINT "y= ";Y: PRINT
2280 PRINT "Vo= ";VO: PRINT
2290 PRINT "t= ";T: PRINT
2300 PRINT "@= ";ANG
2310 GO SUB 1420
2500 DATA "CUAL ES EL MAXIMO ALCANCE 'x' ?"
2510 DATA "DIME LA ALTURA MAXIMA 'y' ?"
2520 DATA "DIME LA VELOCIDAD INICIAL 'Vo' ?"
2530 DATA "DAME EL TIEMPO 't' ?"
2540 DATA "CUAL ES EL ANGULO DE TIRO '@' ?"
2800 PRINT : PRINT
2810 PRINT  FLASH 1;"1";
2820 PRINT ". INCOGNITA 'x' ."
2830 PRINT "2. INCOGNITA 'y' ."
2840 PRINT "3. INCOGNITA 'Vo' ."
2850 PRINT "4. INCOGNITA 't' ."
2860 PRINT "5. INCOGNITA '@' ."
2870 PRINT "6. ELEGIR INCOGNITA."
2880 RETURN
2998 REM
2999 REM **************
3000 REM * GRAVEDADES *
3001 REM **************
```

```
3002 REM
3005 CLS
3010 PRINT AT 0,0; INVERSE 1;"* * GRAVEDADES DEL S. SOLAR. * *"
3020 PRINT
3030 PRINT  BRIGHT 1;"ELIGE EL PLANETA:"
3040 PRINT
3050 LET NOP=9: LET EL=1
3060 PRINT  FLASH 1;"1";
3070 PRINT ". MERCURIO."
3080 PRINT "2. VENUS."
3090 PRINT "3. LA TIERRA."
3100 PRINT "4. MARTE."
3110 PRINT "5. JUPITER."
3120 PRINT "6. SATURNO."
3130 PRINT "7. URANO."
3140 PRINT "8. NEPTUNO."
3150 PRINT "9. ELEGIR PLANETA."
3160 GO SUB 100: IF VAL K$=NOP THEN  GO TO 3200
3170 GO TO 3160
3200 CLS
3210 PRINT "AHORA TU SISTEMA DE REFERENCIA  SE ENCUENTRA EN ";
3220 RESTORE 3900:FI
3250 PRINT  FLASH 1;P$
3260 PRINT
3270 PRINT "EN ESTE PLANETA LA ACELERACION  DE LA GRAVEDAD ES DE"
3280 PRINT "g=";G;" M/S^2"
3290 GO TO 1420
3900 DATA "MERCURIO",4
3901 DATA "VENUS",8.8
3902 DATA "LA TIERRA",9.8
3903 DATA "MARTE",4
3904 DATA "JUPITER",25.4
3905 DATA "SATURNO",10.8
3906 DATA "URANO",8.8
3907 DATA "NEPTUNO",10.8
3998 REM
3999 REM ************
4000 REM * UNIDADES *
4001 REM ************
4010 CLS
4020 PRINT  INVERSE 1;"* * U N I D A D E S * *"
4030 PRINT
4040 PRINT "-'x'. e 'y' en metros."
4050 PRINT
4060 PRINT "-Velocidad inicial 'Vo' en m/s."
4070 PRINT
4080 PRINT "-Tiempo 't' en segundos."
4090 PRINT
4100 PRINT "-Angulo de tiro '@' en radianes."
4110 PRINT
4120 PRINT "-Gravedad 'g' en m/s^2."
4130 GO TO 1420
4998 REM
4999 REM **********************
5000 REM *ECUACIONES GENERALES*
5001 REM **********************
5002 REM
5010 CLS
5020 PRINT  INVERSE 1;"* * E C U A C I O N E S * *"
5030 PRINT : PRINT
5040 PRINT "x=Vo*t*COS @"
5050 PRINT : PRINT
5060 PRINT "y=Vo*t*SIN @ - (g*t^2)/2": PRINT : PRINT
5070 PRINT "Alcance=(Vo^2*SIN (2*@))/g": PRINT : PRINT
5080 PRINT "Alt.(max.)=(Vo^2*(SIN @)^2)/2*g"
5090 GO TO 1420
5100 RESTORE 2520
```

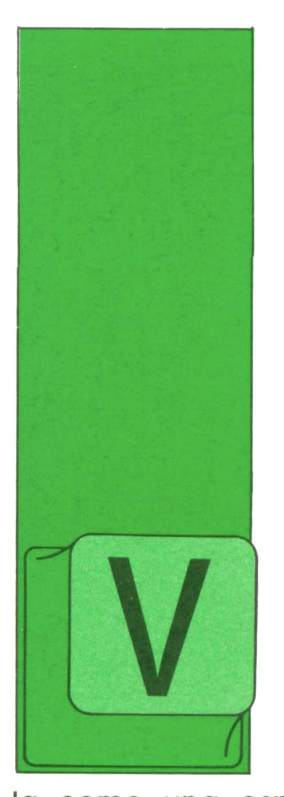
```
5110 READ A$,B$,C$
5120 PRINT A$: INPUT VO
5130 CLS : PRINT B$: INPUT T
5140 CLS : PRINT C$: INPUT ANG
5150 LET X=VAL X$
5160 LET Y=VAL Y$
5170 RETURN
5210 READ A$,B$,B$,C$
5220 PRINT A$: INPUT Y
5230 CLS : PRINT B$: INPUT T
5240 CLS : PRINT C$: INPUT ANG
5250 LET VO=SQR ((2*G*Y)/((SIN ANG)^2))
5260 LET X=VAL X$
5270 RETURN
5300 RESTORE 2510
5310 READ A$,B$,C$,C$
5320 PRINT A$: INPUT Y
5330 CLS : PRINT B$: INPUT VO
5340 CLS : PRINT C$: INPUT ANG
5350 LET T=VO*SIN ANG+SQR (VO^2*SIN ANG^2-2*G*Y)/G
5360 LET X=VAL X$
5370 RETURN
5400 RESTORE 2510
5410 READ A$,B$,C$
5420 PRINT A$: INPUT Y
5430 CLS : PRINT B$: INPUT VO
5440 CLS : PRINT C$: INPUT T
5450 LET ANG=ASN (SQR (2*G*Y/VO^2))
5460 LET X=VAL X$
5470 RETURN
5500 RESTORE 2500
5510 READ A$,B$,B$,B$,C$
5520 PRINT A$: INPUT X
5530 CLS : PRINT B$: INPUT T
5540 CLS : PRINT C$: INPUT ANG
5550 LET VO=SQR (G*X/SIN (2*ANG))
5560 LET Y=VAL Y$
5570 RETURN
5600 RESTORE 2500
5610 READ A$,B$,B$,C$,C$
5620 PRINT A$: INPUT X
5630 CLS : PRINT B$: INPUT VO
5640 CLS : PRINT C$: INPUT ANG
5650 LET T=X/(VO*COS ANG)
5660 LET Y=VAL Y$
5670 RETURN
5700 RESTORE 2500
5710 READ A$,B$,B$,C$
5720 PRINT A$: INPUT X
5730 CLS : PRINT B$: INPUT VO
5740 CLS : PRINT C$: INPUT T
5750 LET ANG=(ASN (G*X/VO^2))/2
5760 LET Y=VAL Y$
5770 RETURN
5800 RESTORE 2500
5810 READ A$,B$,C$,C$,C$
5820 PRINT A$: INPUT X
5830 CLS : PRINT B$: INPUT Y
5840 CLS : PRINT C$: INPUT ANG
5850 LET VO=SQR (G*X/(SIN (2*ANG)))
5860 LET T=X/(VO*COS ANG)
5870 RETURN
5900 RESTORE 2500
5910 READ A$,B$,C$,C$
5920 PRINT A$: INPUT X
5930 CLS : PRINT B$: INPUT Y
5940 CLS : PRINT C$: INPUT T
```

```
5950 LET ANG=ATN ((2*Y+G*T^2)/(2*X))
5960 LET VO=X/(T*COS ANG)
5970 RETURN
6000 RESTORE 2500
6010 READ A$,B$,C$
6020 PRINT A$: INPUT X
6030 CLS : PRINT B$: INPUT Y
6040 CLS : PRINT C$: INPUT VO
6070 LET ANG=(ASN (G*X/VO^2))/2
6080 LET T=X/(VO*COS ANG)
6090 RETURN
```

Para utilizar los menús de opciones que nos aparecen en el programa, sólo tenemos que pulsar el número que aparece junto a la opción que queremos elegir. Una vez hecho esto, dicho número comenzará a parpadear. En ese momento, si nos hemos equivocado, podemos elegir otra opción pulsando otro número distinto. Cuando estemos seguros de que el número que hemos pulsado corresponde a la opción que queremos, debemos pulsar el número que se encuentra junto al mensaje 'GOTO OPCION'. En ese instante, y no antes, dicha opción se ejecutará.

TECNICAS DE ANALISIS

INFORMATIZACION DE LA PRODUCCION INDUSTRIAL

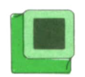


VAMOS a abordar una actividad que está adquiriendo enorme auge en los últimos años: la implantación de sistemas automatizados de producción; incluso se perfila como una especialidad dentro del análisis: la de estudio, diseño e implantación de estos sistemas automatizados de producción.

En algunos aspectos, por otro lado, desborda el campo que se ha considerado típico de los analistas informáticos y de organización: en efecto, además de la informática, de la sistémica y otras ciencias de la organización, el "analista de sistemas automatizados de producción" debe abordar, en ocasiones, actividades propias de la mecánica, la automática y la electrónica.

El conjunto de las técnicas que ha de conocer un analista especializado en sistemas industriales se agrupan, usualmente, bajo las siglas XAO, y son la "concepción asistida por ordenador"-CAO, el "gobierno de máquinas herramientas por ordenador"-CMO, la "gestión de la producción asistida por ordenador"-GPAO, la "fabricación asistida por ordenador"-FAO, el "mantenimiento asistido por ordenador"-MAO, etc.

Aunque ya en los años sesenta comenzaron a informatizarse las operaciones de gestión de la producción (básicamente entonces en cuanto a sus aspectos administrativos y organizacionales), a partir de 1975, con el desarrollo de la microelectrónica y (ya en los ochenta) con la microinformática, el auge de estas técnicas ha sido espectacular, de tal modo que actualmente ya está aceptado el galicismo "prodúctica" para designar a esta "ciencia" específica.



Funciones involucradas en la producción industrial

Las principales actividades englobadas en un proceso genérico de producción se pueden reunir en dos grupos: actividades operacionales (a las que se suele aludir, globalmente, bajo el nombre de "sistema físico de producción"): concepción, aprovisionamiento y fabricación; y funciones de "pilotaje de la producción" o de "gestión de la producción", que permiten el adecuado desarrollo de las anteriores: gobierno, explotación, control y decisión. Veamos brevemente el contenido de estas tareas:

— En el apartado *concepción* hay que incluir, por lo que a la automatización se refiere, las tareas de investigación o estudio (nuevos productos, nuevos materiales, nuevas técnicas de fabricación...), las de desarrollo (estudio de procedimientos, preparación de prototipos o maquetas, etc.), las de "industrialización" o preparación de procedimientos-utillaje-montajes, etc.

— Por *aprovisionamiento* entendemos la adquisición de los materiales (elección de los proveedores, gestión de pedidos, gestión de la posible subcontratación de partes...), transporte (fechas, puntos de aprovisionamiento y de almacenajes transitorios, etc.) y almacena-

miento ("stockage" propiamente dicho y provisión a los puntos en que es necesario cada elemento).

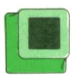
— La *fabricación* comporta, en cuanto actividad automatizable, las siguientes subtareas: "stockage" de elementos y partes como componentes o subconjuntos, a lo largo de todo el proceso productivo; preparación de los elementos necesarios para la fabricación; movimiento de los elementos; transformación y ensamblaje.

— El *gobierno de la producción* incluye todas las tareas de planificación y ejecución de las órdenes de fabricación, compra, movimiento de materiales, etc.

— La *explotación* de los medios de producción supone tanto la conducción de los procesos como el mantenimiento de los medios de producción e, incluso, la formación (en cuanto "mantenimiento" de la "capacidad de producción" de los medios humanos).

— El *control* se realiza tanto desde el punto de vista técnico (control de calidad) como desde el económico y organizativo.

— Las tareas de *decisión* son las más delicadas y constituyen el núcleo del sistema automático de gestión de la producción.



Tipología de los sistemas de producción

El tipo de sistema utilizado en cada producción depende de numerosas razones: de las cantidades fabricadas, de la complejidad del producto, del método de comercialización, de los procedimientos de concepción, etc. Desde el punto de vista de la automatización, se suelen clasificar en los siguientes tipos: en función del sistema físico de producción, como de proceso continuo o discontinuo (y, en este segundo caso, de producción unitaria, repetitiva en series o repetitivo por lotes) y en función de la naturaleza del sistema de pilotaje como de fabricación por stocks, de ensamblaje bajo pedido, de fabricación bajo pedido, etc.



Objetivos de la producción

Las tareas básicas de cualquier sistema de supervisión de la producción (que, por tanto, son el objetivo fundamental que se ha de optimizar con un sistema automatizado) son:

— *Garantizar* el suministro de las mercancías o productos comprometidos por el departamento comercial de la compañía con los clientes.
— *Reducir* los tiempos de disponibilidad de nuevos productos.
— *Ayudar* a la cumplimentación de los pedidos en las mejores condiciones económicas posibles para aumentar la rentabilidad de la empresa.
— *Asegurar* el cumplimiento riguroso de los procedimientos y la coherencia de las informaciones de que se dispone sobre el proceso productivo.

La tan conocida regla de los seis ceros asigna como objetivo a un sistema de producción el conseguir "0 defectos, 0 stock, 0 retrasos, 0 papeles, 0 averías, 0 accidentes".
Perfilando más esta idea, podemos decir que los objetivos a obtener con un sistema de este tipo son:

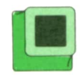
Objetivos técnicos

— Disminuir la duración del ciclo de producción (0 retrasos).
— Mejorar la calidad de los productos, tanto por razones comerciales (para mantener o aumentar la presencia de la compañía en el mercado) como por razones de tipo técnico o financiero.
— Mejorar la disponibilidad del sistema de producción: fiabilidad de los equipos (0 averías), absentismo, política de mantenimiento, etc.
— Mejorar la flexibilidad, para disminuir los retrasos a soportar en caso de modificación de los planes de fabricación, la posibilidad de desarrollar nuevos productos, etc.

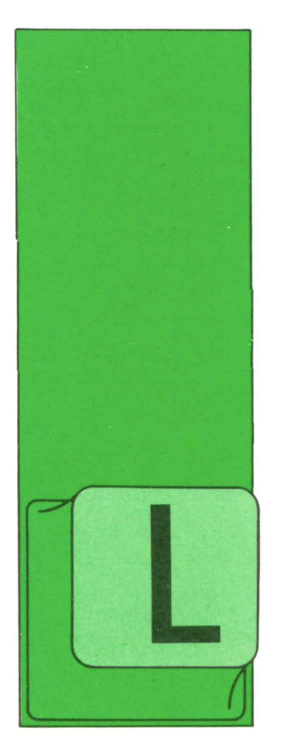
Objetivos económicos

— Disminuir los costes de producción.
— Disminuir el valor de los stocks.
— Asegurar una capacidad óptima de producción.

TECNICAS DE PROGRAMACION



Música por ordenador



LA generación de música por ordenador no sólo es útil para los músicos profesionales o aficionados, sino que cualquier programador puede utilizarla para animar sus programas y mejorar la apariencia externa de éstos, introduciendo pequeñas frases musicales en momentos clave.

Supongamos que se está programando un juego de aventura. Resulta agradable hacer que el ordenador personal (si dispone de un pequeño altavoz) emita en determinadas circunstancias melodías cortas apropiadas a lo que está sucediendo durante el juego. Por ejemplo, si el jugador ha logrado vencer a un enemigo, podrían generarse algunas notas del *Himno a la alegría* de la *Novena sinfonía,* de Beethoven. Si, por el contrario, el enemigo ha "matado" al jugador, no es mala idea hacer sonar algunas notas de la *Marcha fúnebre,* de Chopin.

La forma exacta de generar música en el lenguaje BASIC depende del ordenador personal de que se disponga, y aun en el mismo ordenador pueden existir varias maneras diferentes. La versión que vamos a ver aquí es una de las formas de conseguirlo en el intérprete de BASIC de Microsoft, que funciona, entre otros, en el ordenador personal de IBM y compatibles.

La instrucción básica que permite generar música en BASIC es la siguiente:

PLAY "cadena de caracteres".

La cadena de caracteres que sirve de argumento a la operación PLAY contiene una serie de letras, cifras y puntos que se interpretan de la siguiente manera:

— Las notas musicales aparecen en notación inglesa:

A	LA	E	MI
B	SI	F	FA
C	DO	G	SOL
D	RE	P	Pausa o silencio

El nombre de cada nota puede ir seguido (o no) por el signo + o # (sostenido) o el signo — (bemol), por un número de 1 a 64, que especifica su longitud, y por un punto. Entre las longitudes posibles, destacan las siguientes:

1	Redonda	16	Semicorchea
2	Blanca	32	Fusa
4	Negra	64	Semifusa
8	Corchea		

El punto, si existe, multiplica por 1,5 la duración de la nota anterior.

— Longitud por defecto: la letra L seguida por un número de duración, que se aplicará a todas las notas que no lo tengan.

— Octava: la letra O seguida del número de octava escogido. Se puede elegir entre 7 octavas diferentes, representadas por los números de 0 a 6. Una octava va siempre desde DO (C) hasta SI (B). La octava O0 es la más grave, mientras que O6 es la más aguda.

— Velocidad (tempo): la letra T seguida por el número de notas negras por minuto (un número entre 32 y 255).

— Modo: la letra M seguida por una L (para música ligada), una N (para música normal, con las notas consecutivas ligeramente separadas), o una S (para música "staccato", con las notas muy separadas entre sí).

Veamos algunos ejemplos de música escrita en BASIC:

1. El principio del primer tiempo de la *Sexta sinfonía,* de Beethoven.

```
10 PLAY "T9003L8EFAGL16FEL8D02G03CDEF16E16D4."
```

2. El principio del último tiempo de la *Sexta sinfonía,* de Beethoven.

```
10 PLAY "T9003L8CA4.F404C03A4.F4CF4A04C403AB-2."
20 PLAY "AB-4.G404C03A4.F4AG4DE4CCF4."
```

3. El principio del *Himno a la alegría,* del último tiempo de la *Novena sinfonía,* de Beethoven.

```
10 PLAY "T9003L8EEFGGFEDCCDEE.D16D4"
20 PLAY "EEFGGFEDCCDED.C16C4"
```

4. El principio de la *Marcha fúnebre,* de Chopin.

```
10 PLAY "T5002L8A4A.A16A403C.02B16B.A16A.A16A2"
```

Se comprende la facilidad con que pueden construirse programas que ejecuten piezas musicales mucho más complicadas.



Música en APL

Como en los capítulos anteriores, nos referiremos al intérprete de APL/PC de IBM. Otros intérpretes de APL pueden realizar la música de manera diferente, o incluso no estar dotados de esta posibilidad.

Este intérprete de APL se distribuye con una función (PLAY) que actúa de forma muy semejante a la instrucción PLAY de BASIC, aunque con algunas diferencias.

En primer lugar, la cadena de caracteres a la que se aplica la función PLAY debe estar encerrada entre comillas simples, como siempre ocurre en APL. En segundo lugar, veamos cómo se interpretan aquí las letras y cifras contenidas en dicha cadena de caracteres:

— Las notas musicales aparecen en notación inglesa:

A	LA	E	MI
B	SI	F	FA
C	DO	G	SOL
D	RE	P	Pausa o silencio

El nombre de cada nota puede ir seguido (o no) por el signo + o # (sostenido) o el signo — (bemol), por un número de 0 a 6, que especifica su longitud, y por un punto. Las longitudes posibles son las siguientes:

0	Redonda	4	Semicorchea
1	Blanca	5	Fusa
2	Negra	6	Semifusa
3	Corchea		

El punto, si existe, multiplica por 1,5 la duración de la nota anterior.

— Longitud por defecto: la letra L seguida por un número de duración, que se aplicará a todas las notas que no lo tengan.

— Octava: la letra O seguida del número de octava escogido. Se puede elegir entre 7 octavas diferentes, representadas por los números de 0 a 6. Una octava va siempre desde DO (C) hasta SI (B). La octava O0 es la más grave, mientras que O6 es la más aguda. Si el número de octava va seguido por un signo +, como en O1+, significa que se suben esas octavas respecto a la octava anteriormente definida. Si va seguido por un signo —, como en O1—, significa que pasamos a una octava más baja (más grave) respecto a la anterior.

— Velocidad (tempo): la letra T seguida por un número comprendido entre 0 y 6, de acuerdo con la siguiente tabla:

0	Largo	(54 notas negras por minuto)
1	Largetto	(66 notas por minuto)
2	Adagio	(78 notas por minuto)
3	Andante	(96 notas por minuto)
4	Moderato	(120 notas por minuto)

5 Allegro (156 notas por minuto)
6 Presto (198 notas por minuto)

— Modo: la letra M seguida por un 2 (para música ligada), un 1 (para música normal, con las notas consecutivas ligeramente separadas), o un 0 (para música "staccato", con las notas muy separadas entre sí).

Veamos cómo se escriben en APL los ejemplos dados anteriormente en BASIC:

1. El principio del primer tiempo de la *Sexta sinfonía*, de Beethoven.

```
PLAY 'T3O3L3EFAGL4FEL3DO2GO3CDEF4E4D2.'
```

2. El principio del último tiempo de la *Sexta sinfonía*, de Beethoven.

```
PLAY 'T3O3L3CA2.F2O4CO3A2.F2CF2AO4C2O3AB-1.'
PLAY 'AB-2.G2O4CO3A2.F2AG2DE2CCF2.'
```

3. El principio del *Himno a la alegría*, del último tiempo de la *Novena sinfonía*, de Beethoven.

```
PLAY 'T3O3L3EEFGGFEDCCDEE.D4D2'
PLAY 'EEFGGFEDCCDED.C4C2'
```

4. El principio de la *Marcha fúnebre*, de Chopin.

```
PLAY 'TOO2L3A2A.A4A2O3C.O2B4B.A4A.A4A1'
```

Obsérvese que la conversión de un lenguaje al otro es trivial.

Naturalmente, varias llamadas a la función PLAY pueden combinarse para formar un programa o función de APL independiente, que puede permitir construir piezas musicales realmente complicadas sin demasiado esfuerzo y en muy pocas líneas. Veamos algunos ejemplos, por orden de complejidad creciente:

1. Una pieza musical compuesta por mi hijo Enrique cuando tenía cinco años de edad.

```
[0]  CONCITO
[1]   PLAY 'M2O3L2T5'
[2]   PLAY 'GO4ECO3P3G3AG3F3GE'
[3]   PLAY 'GO4ECO3P3G3AG3F3G1'
[4]   PLAY 'GO4ECO3P3G3AG3F3GE'
[5]   PLAY 'GGGP3G3AG3G3C1'
```

2. Una pieza musical compuesta por mi hija María de los Angeles cuando tenía siete años de edad.

```
[0]  MARCHA
[1]   PLAY 'M2O3L2T5'
[2]   PLAY 'C3EC3O2GGO3D3FD3O2G1'
[3]   PLAY 'O3C3EC3O2GGG3G3A3B3O3C1'
[4]   PLAY 'C3EC3O2GGO3D3FD3O2G1'
[5]   PLAY 'O3E3GE3O2GGG3O3CO2B3O3C1'
```

3. El *Himno nacional* español.

```
[0]  HIMNO
[1]   PLAY 'O2L2T3M1O3CO2GO3EL3CGFEDCCO2BAG'
[2]   PLAY 'O3L2CDE.L3GFEDCG1G2EGF2DFE2CED'
[3]   PLAY 'O2GABO3L2CDL3EFGFL2EDC1P1'
```

4. Un fragmento de *Ensueño* (o *Reverie*), de Roberto Schumann.

```
[0]  REVERIE
[1]   PLAY 'M2T303L3'
[2]   A←'F1FEFA04CM1FM2F1EDCF03GAB-04D03FGAB+G1C2F1FEFA04CM1AM2A2.GF'
[3]   B←'EFADFECCE-D1E.M1C4M2C2.03C'
[4]   PLAY 'C2',A,B,A,B
[5]   PLAY 'F1FEFA04CM1E-E-1M2DC03B-04D03GAB-GGAG2.M1DM2D2PF'
[6]   PLAY 'B-1B-AB-04DFM1B-B-1M2AGFADEFDDED03AM1AM2AA2G2'
[7]   PLAY A,'DCF03GAB-04D03GAB-04D03DEF1P0'
```

5. Un fragmento de *Para Elisa*, de Beethoven.

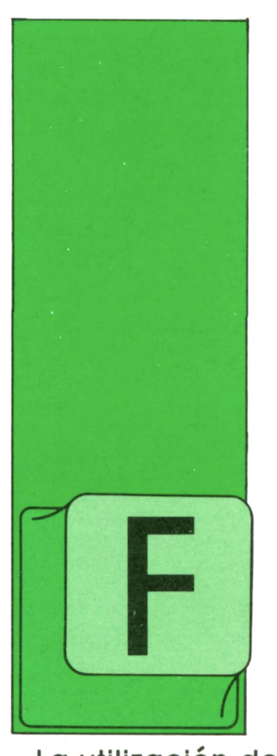
```
[0]  ELISA
[1]   S←0
[2]   PLAY 'T4M1'
[3]  L:PLAY EL1←'04L3ED+ED+E03B04DC03A2PCEAB2P'
[4]   PLAY 'EG+B04C2P03E'
[5]   PLAY EL1,'E04C03BA2P'
[6]   →(~S←~S)/0
[7]   PLAY 'B04CDE2.03G04FED2.03F04EDC2.03E04DC03B2EE04E03E04E2.D+'
[8]   →L
```

6. El vals de *La Bella Durmiente del Bosque*, de Chekovski. La ejecución de este programa dura, aproximadamente, cinco minutos.

```
[0]  DURMIENTE
[1]   SW←1
[2]   PLAY 'T6L2M1',A,A←'02C03EE01G03EE'
[3]  L:A←'M204C1.03B1.04C103ABB+AB104DE1C+D0.G1.F+1.F1D'
[4]   PLAY A←A,'FEDA1GF1M1EM2EDC+D03AB04C1.03B1.04C103A'
[5]   PLAY 'BB+AB104M1DM2D1C+D0.L1EFF+AGBB+BAL3GPF+GBAGEDED03BG'
[6]   B←'P04G1A1B2B+PD+EFEGPC+DEDEPC-CDCEDCC-C03AGP04F+GAGBAGF+GD+EP'
[7]   PLAY B,B←B,'G1A1B2B+PD+EFEGPC+DEDEPC-CDCEDC03BAGF+P04E+F+GF+BAGF+EDG'
[8]   PLAY 'FEDEDL2',A
[9]   PLAY 'B04CC+D1C-E1C+L3FP203C+DFAG+A04C+DFL1G+AB05DCEFEDL3CP'
[10]  →(SW←~SW)/L1
[11]  PLAY A←'P1.04G205D1.04G1G2B+1.G1L2EGDAGDAGD+EC-C'
[12]  PLAY B←'G05D1.04G1GB+1.G1ABAGF+B+.B3E1.'
[13]  PLAY A,B,'D1.'
[14]  →L
[15] L1:PLAY '04BB+GECP03BB+GEC2P1EPP1CPP1'
```

La conversión de estos programas a otros equivalentes escritos en BASIC es muy sencilla. Obsérvese que L1 debe transformarse en L2, L2 en L4, L3 en L8, L4 en L16, T3 en T96, T4 en T120, T5 en T156, T6 en T198, M0 en MS, M1 en MN y M2 en ML. Por último, la conversión de las longitudes de las notas es idéntica a la de Ln.

APLICACIONES





FRAMEWORK

FRAMEWORK constituye un paquete integrado que incluye un procesador de textos, una base de datos, una hoja electrónica, gráficos y comunicaciones.

La utilización de este paquete se realiza a través de ventanas con las cuales se pueden ver los esquemas, como podremos ver más adelante.

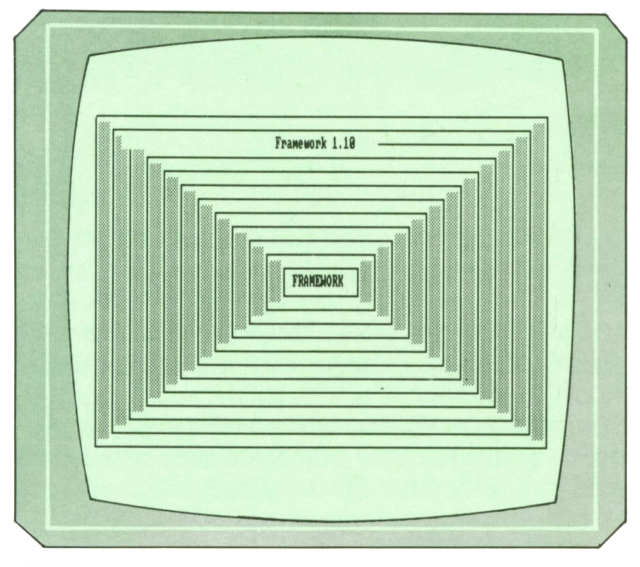




Fig. 1. Pantalla de presentación de FRAMEWORK.

La gran ventaja de un paquete integrado de este tipo es la posibilidad de utilizar los ficheros por todos los módulos, lo que evita la repetición de los datos. Existen tres formas posibles de comunicar los datos de un módulo a otro.

La primera consiste en ir abriendo ventanas en las que se van incluyendo los datos proporcionados por cada uno de los módulos. La segunda posibilidad es referenciar los datos de la hoja electrónica o de la base de datos por el nombre de fichero y el número de archivo que ocupa el dato, de esta forma cualquier cambio que se efectuase en la hoja o en la base de datos se transformará automáticamente al sitio donde se mandó.

La última manera es a través de los esquemas; ésta consiste en una serie de ventanas que llamarán a una serie de otras ventanas, produciendo estructuras parecidas a las llamadas de árbol de las bases de datos jerárquicas.

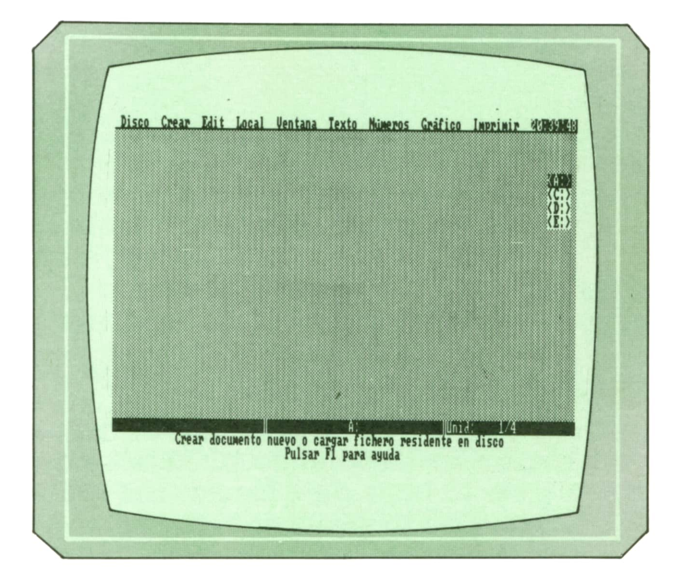




Fig. 2. Pantalla de trabajo de FRAMEWORK.

Las ventanas de nivel más interno pueden ser cualquier tipo. Aunque teóricamente el número de niveles puede ser ilimitado, en la práctica queda restringido por la capacidad de memoria del ordenador que se esté utilizando.



Tratamiento de textos

En FRAMEWORK la aplicación central la constituye el procesador de textos. El que incluye este paquete es de gran potencia y además de poseer todas aquellas funciones del procesador de textos habituales, tales como centrado de texto, modificación y supresión. Incluye subrayados, alineación a la derecha, etc. Sin embargo, la longitud máxima de los ficheros queda reducida a la capacidad de memoria del ordenador, ya que los documentos se almacenan enteros en la memoria RAM.

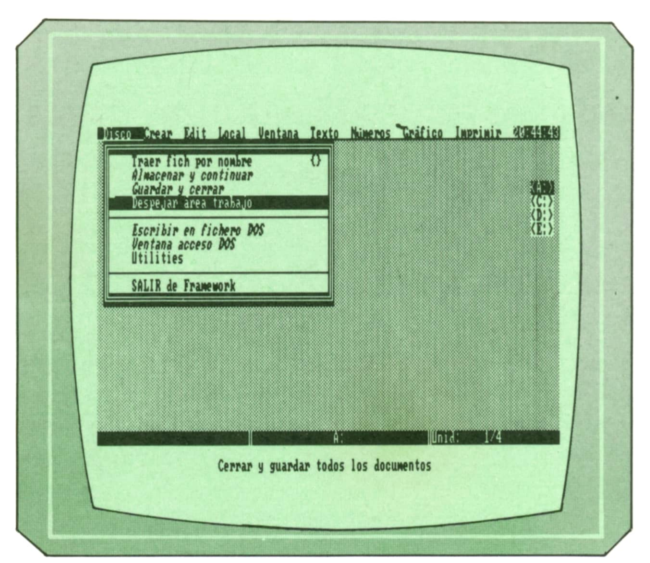




Fig. 3. Opciones de manejo de discos.



Gráficos

Al módulo de creación de gráficos se accede desde el submenú gráficos. Estos gráficos se generan a partir de los datos contenidos en los ficheros de la base de datos y de la hoja de cálculo. Los gráficos son de tipo empresarial y pueden ser de tipo: de barras, de pastel, de líneas, de puntos y de ejes XY. Pueden superponerse gráficos diferentes y realizarlos con más de un fichero, así como cambiar una escala.



Base de datos

La base de datos permite el mantenimiento de los ficheros necesarios. Para crear uno de ellos es necesario suministrar al ordenador el número de registros que va a contener. No permite la creación de índices, ya que el fichero está contenido en la memoria central, con lo que su acceso es siempre directo.

A los ficheros de la base de datos se le pueden añadir campos y registros, pudiendo relacionarse aquellos registros que interesen mediante la creación de filtros; éstos pueden ordenarse tanto ascendentemente como descendentemente.

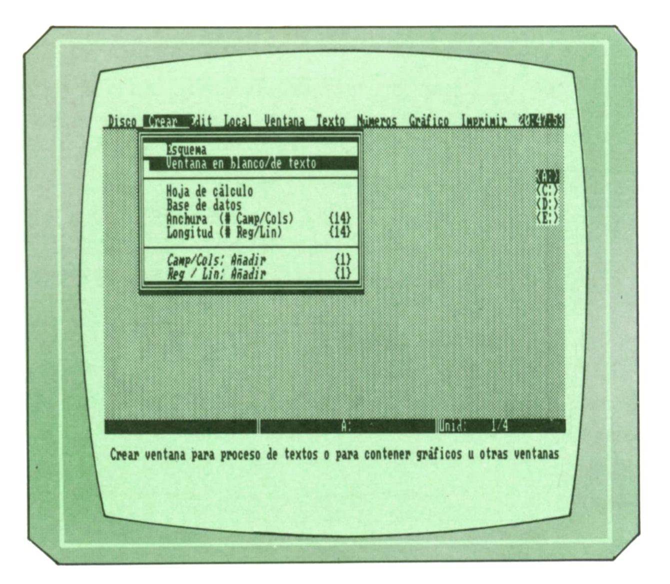




Fig. 4. Ventana de crear.



Hoja electrónica

Para trabajar con la hoja electrónica hay que especificar primero su tamaño, filas y columnas.

A cada celdilla o grupos de ellas es posible etiquetarlos poniéndoles un nombre, opción muy útil cuando hace de referencia posteriormente.

Este módulo incluye numerosas funciones matemáticas, estadísticas, financieras y trigonométricas, etc.

Una de las características más interesante es la de relacionar datos procedentes de otras hojas de cálculo, realizando operaciones entre células procedentes de diversas hojas de cálculo. Una de las ventajas importantes de esta hoja electrónica es que permite la utilización de gran número de los comandos de procesos de textos.

Comunicaciones

El módulo de comunicaciones permite conectar el ordenador con las bases de datos remotas, así como enviar y recibir información, en forma de ficheros de todo tipo, tanto de texto como de cualquier otro. Asimismo tiene capacidad para realizar la emulación de terminales. Puede manejar distintos tipos de modems conectados a una línea telefónica.

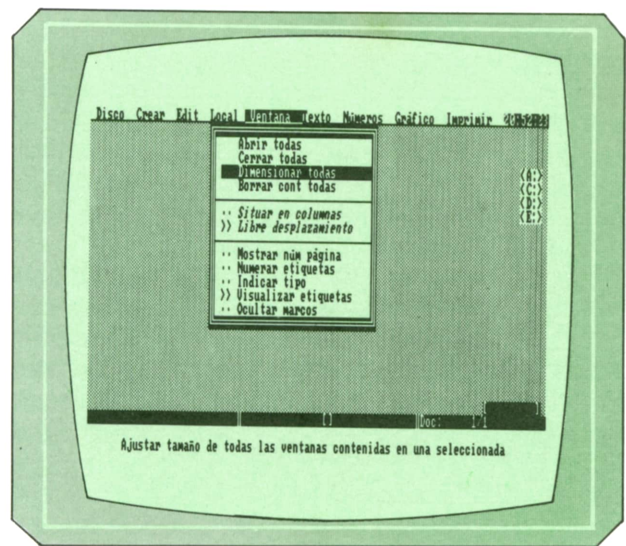




Fig. 5. Opciones de las ventanas.



Lenguaje FRED

A pesar de que todo lo anteriormente visto ya indica la gran potencia de este paquete cuenta con un elemento más, que es lo que realmente le otorga una flexibilidad de manejo impresionante, el lenguaje de programación FRED.

Este lenguaje, utilizable para cualquier aplicación de FRAMEWORK II, incluye funciones para trabajos financieros y matemáticos, formateo y conversión de números, formateo y tratamiento de textos, conversiones de formatos numéricos y de fechas, manejo de caracteres, pantallas, ventanas y menús, un control completo de las secuencias de ejecución de los programas, así como el soporte de las funciones y aplicaciones generadas por el usuario.

Los programas y las fórmulas de líneas múltiples pueden contener hasta 64.000 caracteres de longitud, pudiéndose además insertar comentarios, bien en una línea independiente o al final de una sentencia ejecutable.

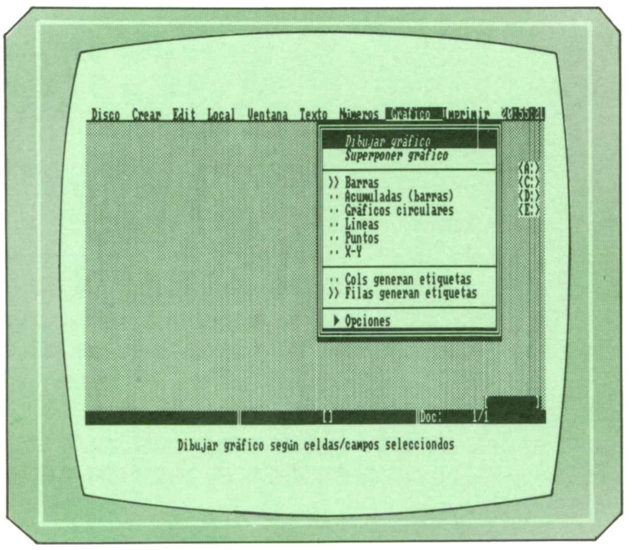




Fig. 6. Opciones gráficas.

Ya que cada ventana puede tener una fórmula, los programas FRED se pueden aplicar desde los documentos, hojas de cálculo, bases de datos, gráficos y comunicaciones.

También se pueden pasar la información y el control de unos a otros, lo que unido al anidamiento de ventanas, permite la construcción de cualquier tipo de aplicación para cubrir las necesidades del usuario.

Realmente todas las funciones del FRAMEWORK son fórmulas realizadas en FRED, de forma que mediante este lenguaje se puede acceder a cualquier operación de FRAMEWORK.

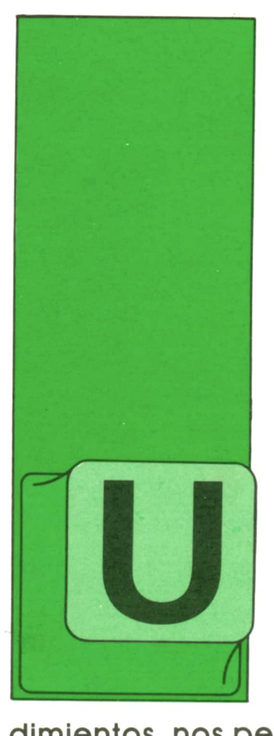
Es imposible pasar revista a todas las funciones del FRED. Sin embargo, y para dar una pequeña muestra de su potencia, he aquí alguna de sus peculiaridades:

— Manejo de variables locales.
— Funciones de construcción de funciones.
— Funciones de referencia y manejo de ventanas.
— Sentencias de bifurcación y reiteración.
— Funciones de interfaz de usuario.
— Generación de macroinstrucciones.
— Funciones de formato e impresión.
— Manejo de series de datos.

En definitiva, FRED ofrece una versatilidad difícil de igualar, tanto por la potencia de sus funciones como por la flexibilidad de manejo.

PASCAL

El Pascal y las matemáticas (2)



NA vez visto cómo programar las operaciones principales que se pueden realizar con polinomios, vamos a desarrollar un programa que, por medio de esos procedimientos, nos permita obtener las raíces reales de un polinomio cualquiera. Para ello utilizaremos el denominado método de Sturm.



El método de Sturm

El método (que no justificaremos) consiste en lo siguiente:

A partir del polinomio cuyas raíces queremos calcular se genera un conjunto de polinomios denominado "secuencia de Sturm". Si guardamos esa secuencia en una tabla de polinomios (a la que llamaremos Sec), la forma de generarla sería la siguiente:

Sec(0) el polinomio original.
Sec(1) la derivación de Sec(0).
Sec(2) el resto de la división de Sec(0) entre Sec(1), cambiado de signo.
Sec(3) el resto de la división de Sec(1) entre Sec(2), cambiado de signo.

Y así hasta llegar a Sec(N), donde N sería el grado del polinomio original, es decir, de Sec(0)

Supongamos que el grado fuese 3; si en un punto dado evaluásemos por orden los polinomios de la secuencia y nos fijásemos únicamente en el signo de cada resultado, podríamos llevar la cuenta de los cambios de signo que se fuesen produciendo hasta llegar a Sec(3). Por ejemplo, si los signos de Sec(0), Sec(1), Sec(2) y Sec(3) fuesen, respectivamente, más, más, menos y más, el número de cambios de signo sería 2 (el paso de más a menos, y el de menos a más).

Pues bien, resulta que si evaluamos el número de cambios de signo de la secuencia de Sturm de un polinomio dado en dos puntos distintos, la diferencia entre ambos resultados nos indica la cantidad de raíces reales del polinomio que se encuentran entre esos puntos. En concreto, la diferencia en los números de cambios de signo que se producen en menos infinito y más infinito indica el total de raíces reales que existen. El número de cambios de signo en el menor de los dos puntos es siempre mayor o igual que en el otro.

Por tanto, el método consiste en, una vez que se ha determinado un intervalo en el que existe alguna raíz real, ir estrechando el cerco en torno a ella hasta determinarla.

Veamos un ejemplo. Llamemos C(x) a una función que nos indique el número de cambios de signo en X; si C(—infinito) vale 3 y C(+infinito) vale 2, eso quiere decir que hay una única raíz real. Si a continuación evaluamos C(0) y vale 3, eso quiere decir que la raíz está entre 0 y +infinito; si evaluamos C(100) y vale 2, la raíz está, por tanto, entre 0 y 100, etc.

Excepto cuando el grado es cero, no resulta posible evaluar un polinomio en más o menos infinito, aunque sí resulta posible conocer el signo del resultado sin más que observar su grado y el coeficiente del término de mayor grado.

En más infinito, el signo del resultado (que en valor absoluto es infinito) coincide con el del coeficiente. En menos infi-

nito sucede lo mismo si el grado es par, y es al contrario si el grado es impar.

Veamos cómo llevar a la práctica todo esto:

```
program Sturm;
{--------------------------------------------------------}
{  Facilita la obtención de las raíces reales de un      }
{  polinomio cualquiera por medio del m[etodo de Sturm   }
{--------------------------------------------------------}

const
 Grado_Max = 15;

type
 Grado_t      = 0..Grado_Max;
 Polinomio_t = record
                Coef : array [Grado_t] of real;
                Grado: Grado_t
               end;
 Secuencia_t = array [Grado_t] of Polinomio_t;

var
 Pol: Polinomio_t;  { para el polinomio cuyas raíces se van a obtener }
 Sec: Secuencia_t;  { para la secuencia de Sturm asociada }
 X  : real;
 Ca : char;
 I  : integer;

{--------------------------------------------------------------------}
function VALOR (var P: Polinomio_t; X: real): real;
{------------------------------------}
{  Da el valor del polinomio P en X  }
{------------------------------------}
 var Acumula: real; I: integer;
begin
 Acumula := 0.0;
 with P do
   for I := Grado downto 0 do
     Acumula := Acumula * X + Coef [i];
 Valor := Acumula
end;

{--------------------------------------------------------------------}
procedure DERIVAR (var P, D: Polinomio_t);
{-----------------------------------}
{  Hace D igual a la derivada de P  }
{-----------------------------------}
  var I: integer;
begin
with P do
  begin
  for I := 1 to Grado do
    D.Coef [i-1] := Coef [i] * I;
  D.Grado := Grado - 1
  end
end;

{--------------------------------------------------------------------}
procedure NORMALIZA (var P: Polinomio_t);
{------------------------------------}
{  Si el primer coeficiente es nulo  }
{  reduce Grado hasta que no lo sea  }
{------------------------------------}
  var Parar: boolean;
begin
 Parar := false;
 with P do
 while (Grado >= 0) and not Parar do
   if Coef [Grado] <> 0.0 then Parar := true
                          else Grado := Grado - 1
end;

{--------------------------------------------------------------------}
procedure DIVIDIR (var N, D, C, R: Polinomio_t);
{----------------------------------}
{  Hace C y R igual al cociente y  }
{  al resto de dividir N entre D.  }
{----------------------------------}
 var K: real; I, Dif: integer;
begin
Normaliza (D);    { Por si su primer coeficiente es 0. }
```

```
if D.Grado >= 0 then  { Si es un polinomio no nulo... }
    begin
    R := N;
    C.Grado := N.Grado - D.Grado;
    with R do
     while Grado >= D.Grado do
      begin
       { exponente y coeficiente a utilizar:  }
       Dif := Grado - D.Grado;
       K   := Coef [Grado] / D.Coef [D.Grado];

       { Resto := Resto - (K * X^Dif) * Divisor:  }
       for I := Grado - 1 downto Dif do
            Coef[i]:= Coef[i] - K * D.Coef[i-dif];
       Grado := Grado - 1;

       { el término empleado corresponde al cociente: }
       C.Coef [Dif] := K
      end
    end
  else
    begin
     writeln;
     writeln ('Error:  División por polinomio nulo.');
     writeln ('Pulse Intro para seguir.');
     readln
    end
 end;

{------------------------------------------------------------------}
procedure LEE_POL (var P: Polinomio_t; Letra: char);
{-----------------------------------------------------------}
{ Pide grado y coeficientes de P; saca Letra[ ] de pregunta }
{-----------------------------------------------------------}
  var I: integer;
begin
 write ('Grado = '); readln (I); writeln;
 if (I <= Grado_Max) and (I > 0) then with P do
   begin
   Grado := I;
   for I := Grado downto 0 do
      begin
      write (Letra,'[',I,']= ');
      readln (Coef [i])
      end;
   Normaliza (P);
   end
 end;

{------------------------------------------------------------------}
procedure SACA_POL (P: Polinomio_t; Letra: char);
{-----------------------------------------------------------}
{ Saca grado y coeficientes de P; saca Letra[ ] de pregunta }
{-----------------------------------------------------------}
  var I: integer;
begin
 with P do
   for I:=Grado downto 0 do
     writeln (Letra,'[',I,']= ', Coef [i])
end;

{------------------------------------------------------------------}
procedure OBTENER_SECUENCIA (P: Polinomio_t; var S: Secuencia_t);
{------------------------------------------------}
{  Hace F igual a una secuencia de Sturm para P.  }
{  El cociente de las divisiones no se utiliza.   }
{------------------------------------------------}
var
  Cociente: Polinomio_t;
  I, J    : integer;
begin
  S [0] := P;
  Derivar (P, S [1]);
  {------------------------}
  for I := 2 to P.Grado do
    begin
    Dividir (S [i-2], S [i-1], Cociente, S[i]);
    {------------------------------------------}
    {  Cambio del signo de los coeficientes:  }
    {------------------------------------------}
    with S[i] do for J := 0 to Grado do Coef[j] := - Coef[j]
    end
end;
```

```
{----------------------------------------------------------------------}
function CAMBIOS_DE_SIGNO (var S: Secuencia_t; X: real): integer;
{---------------------------------------------------}
{  Devuelve el número de cambios de signo de S en X  }
{---------------------------------------------------}
var
  Signo_Previo, Y : real;
  I, Cambios      : integer;
begin
 { búsqueda del primer polinomio con valor no nulo en X: }
 I := -1;
 repeat
   I := I + 1;
   Y := Valor (S[i], X);
 until Y <> 0.0;
 {----------------------------}
 Signo_Previo:= Y / abs(Y);    { Se guarda +1 o -1, según el signo }
 Cambios     := 0;
 {----------------------------}
 for I := I + 1 to S[0].Grado do       { sequir con los restantes: }
   begin
    Y := Valor (S[i], X);
                                 { Si se produce un cambio de signo: }
    if Signo_Previo * Y < 0.0 then
       begin
        Cambios     := Cambios + 1;
        Signo_Previo:= Y / abs(Y)
       end
   end;
 {----------------------------}
 Cambios_de_Signo := Cambios
 end;


{----------------------------------------------------------------------}
function CAMBIOS_EN_INFINITO (var S: Secuencia_t; Tipo: integer): integer;
{-------------------------------------------------}
{  Devuelve el número de cambios de signo de S en  }
{  + infinito (Tipo = +1) y -infinito (Tipo = -1)  }
{-------------------------------------------------}
var
  Signo_Previo, Y : real;
  I, Cambios      : integer;
begin
 Cambios := 0;
 with S[0] do
   begin
    Y := Coef [Grado];
    if odd (Grado) and (Tipo = -1) then Y := -Y;
    Signo_Previo := Y / abs(Y)       { presupone S[0] normalizado }
   end;
 {----------------------------}
 for I := 1 to S[0].Grado do with S[i] do
   begin
    if Grado < 0 then Y := 0.0    { polinomio nulo... }
                 else Y := Coef [Grado];
    if odd (Grado) and (Tipo = -1) then Y := -Y;
                                 { Si se produce un cambio de signo: }
    if Signo_Previo * Y < 0.0 then
       begin
        Cambios     := Cambios + 1;
        Signo_Previo:= Y / abs(Y)
       end
   end;
 {----------------------------}
 Cambios_en_Infinito := Cambios
end;

{======================================================================}
{                          PROGRAMA PRINCIPAL                          }
{======================================================================}

begin
 repeat
  writeln;
  writeln ('1 - Introducir polinomio.');
  writeln ('2 - Mostrar Secuencia de Sturm.');
  writeln ('3 - Búsqueda de raíces.');
  writeln ('4 - Salida del programa.');
  writeln;
  write ('ESCOJA OPCION: '); readln (Ca);
  writeln;

  case Ca of
```

```
'1': begin                              { entrada de datos }
      Lee_Pol (Pol, 'P');
      Obtener_Secuencia (Pol, Sec)
     end;

'2': for I := 0 to Pol.Grado do      { mostrar secuencia de Sturm }
       begin
        writeln ('------------S[',I,']-------------');
        Saca_Pol (Sec[i], 'S');
        writeln;
        write ('Pulse Intro para seguir'); readln
       end;

'3': begin                           { Búsqueda de raíces }
      writeln ('Número de cambios de signo en - infinito = ',
                Cambios_en_Infinito (Sec, -1));
      writeln ('Número de cambios de signo en + infinito = ',
                Cambios_en_Infinito (Sec, +1));
      repeat
        writeln;
        write ('X   = '); read (X);
        for I:=0 to Pol.Grado do
          begin
           writeln;
           write ('S[',i,']= ',Valor (Sec[i], X))
          end;
        writeln ('         NUMERO DE CAMBIOS DE SIGNO = ',
                  Cambios_de_Signo (Sec, X));
        writeln;
        write ('¿Desea seguir? (S/N) '); readln (Ca);
      until (Ca = 'N') or (Ca = 'n')
     end

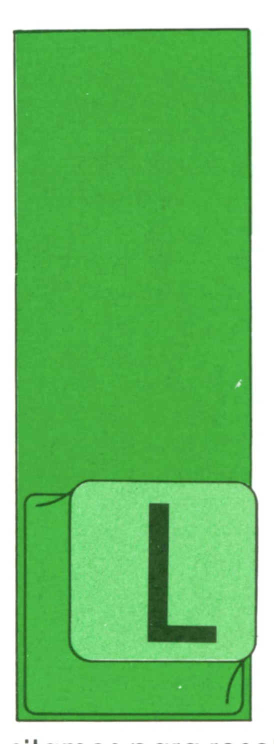
end

until Ca = '4'

end.
```

OTROS LENGUAJES

(PROLOG - 1)



A mayoría de los lenguajes de programación consisten en definiciones, más o menos formales y precisas, de un algoritmo. En ellos se indica qué clase de datos necesitamos para resolver nuestro problema y qué operaciones debemos realizar con ellos para obtener el resultado deseado. Por tanto, sólo sirven para resolver, de forma automática, un problema cuya solución general conocemos con anterioridad.

Esto no es así en PROLOG; en él definimos objetos y relaciones o propiedades de éstos. De estas relaciones el ordenador será capaz de indicarnos nuevas relaciones existentes, o qué objetos tienen determinadas propiedades, pudiendo resolver problemas cuya solución genérica no nos es conocida previamente.

De esta propiedad proviene el nombre PROLOG (PROgramación LOGica) para distinguirlo de la tradicional programación algorítmica. Fue inventado por Alain Colmenaeur en 1970.

Hay que tener en cuenta que debemos conocer algo de cálculo proposicional, o de lógica formal, ya que de lo contrario no sabríamos qué le estamos indicando que haga.

La aplicación principal es la creación de bases de datos más o menos inteligentes y la llamada "inteligencia artificial"; dentro de ésta suele ser utilizado para comprensión del lenguaje natural (el hablado por los seres humanos normalmente), resolución de problemas e implementación de sistemas expertos.

Por sus especiales características es poco usado para las aplicaciones más típicas de los ordenadores hasta ahora, como cálculo de contabilidades o cálculo científico, ya que para esto existen lenguajes más específicos.

Otra diferencia con los lenguajes más convencionales es que no escribimos de golpe el programa en un editor y después lo compilamos. Es un intérprete al que le vamos proporcionando hechos y reglas relacionadas con éstos, manteniendo una especie de diálogo.

Si le indicamos un hecho o una regla él la añadirá a su base de conocimientos para aplicarla más tarde. Si le preguntamos algo, él nos indicará si con los conocimientos que posee esa proposición es verdadera o falsa. También le podemos preguntar qué objetos, de los que él conoce, cumplen una determinada propiedad.

El final de cada sentencia que le digamos debe terminar en un punto ("."). La función de esto es que si una sentencia ocupa más de una línea el intérprete de PROLOG espera a tenerla completa para empezar a interpretarla.

Por ser todos los intérpretes muy lentos, al estar los ordenadores diseñados para ejecutar óptimamente programas algorítmicos, debemos no olvidarnos del punto final, pues podemos creer que está interpretando nuestra sentencia, cuando, en realidad, está esperando el punto de terminación.

Otro error, muy corriente, que debemos evitar es el de escribir un nombre de dos formas distintas; al distinguir entre mayúsculas y minúsculas podemos creer que es el mismo nombre, cuando él lo considera distinto, con los posibles errores fatales que ello puede conllevar.

Un objeto existe con que, sencillamente, lo nombremos; no es necesario que los definamos previamente.

Un hecho es la indicación de una propiedad de un objeto, o de una relación entre objetos.

Veamos un ejemplo:

Para declarar "Pedro tiene un libro", en PROLOG se escribiría, tiene (Pedro, libro).

En esta declaración observamos que se indica primero la relación, o predicado, y después los objetos que la verifican, o argumentos; en este caso sería un predicado con dos argumentos.

También podemos observar que el orden de la colocación de los objetos es importante, ya que:

tiene (libro, Pedro).

No significaría lo mismo, pues indicaría que Pedro es poseído por un libro, lo que no tiene mucho que ver con lo que queríamos declarar.

Según la versión de PROLOG que utilicemos, los hechos pueden escribirse de maneras ligeramente distintas. Como se observará, cada hecho es una lista (ver capítulo dedicado al lenguaje LISP para la noción de lista); por tanto, puede expresarse de diversas formas según la notación aceptada.

El PROLOG no distingue entre lo que pueda significar los datos que posee, sólo los almacena en su base de datos para usarlos posteriormente, de ahí que el significado de los nombres de los objetos y las relaciones debemos asignarlos subjetivamente para hacerlos lo más inteligibles posible; teniendo en cuenta las posibles arbitrariedades, pueden causar en cualquier persona que lee el programa graves problemas de comprensión. Por parte del ordenador no habría problemas en expresar la anterior relación como:

a (b, c).

PREGUNTAS

Una vez que se han definido algunos hechos podemos hacer preguntas sobre ellos. En PROLOG las preguntas son como cualquier hecho, sólo que llevan delante el símbolo de pregunta, una interrogación y un guión "?-", y detrás de ellas un hecho. El intérprete analizará el hecho y nos dirá si es verdadero o falso, respondiéndonos sí o no (*yes* o no, por estar normalmente en inglés).

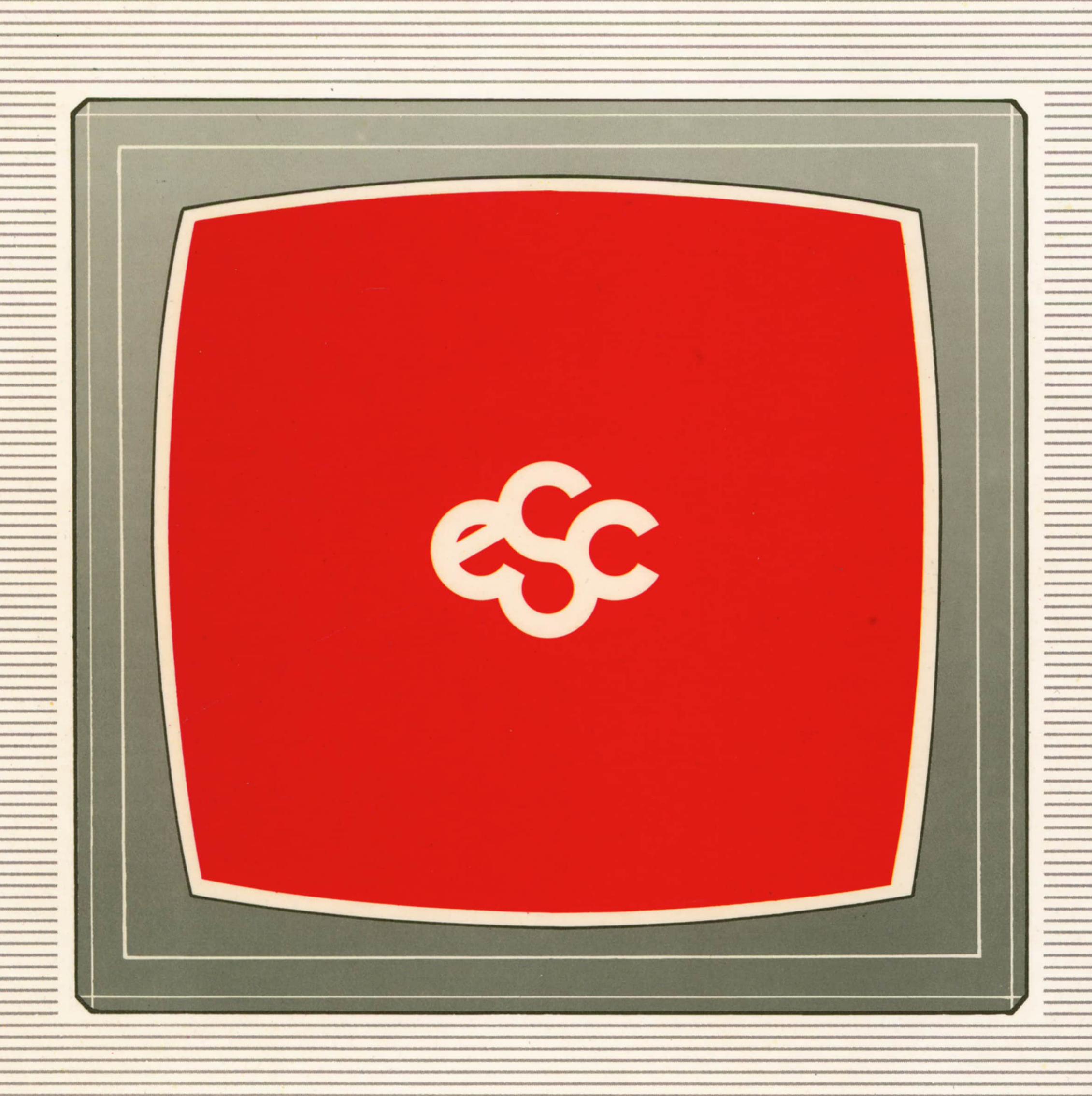
Veamos algunos ejemplos:

Supongamos la siguiente base de conocimientos:

posee (pedro, libro).
posee (pedro, manzana).
posee (juan, pera).
posee (juan, dinero).

Nos respondería a las siguientes preguntas:

?-posee (pedro, pera).
No.
?-posee (juan, libro).
No.
?-posee (pedro, manzana).
No.
?-posee (juan, pera).
Yes.



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼